NONINFERIOR SET SCENARIO ANALYSIS

A THESIS
SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE DEGREE
OF
DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH
IN THE
UNIVERSITY OF CANTERBURY
by

Oliver Colin McCahon

University of Canterbury
2000

To Dayll, and to Lucy, Walter and Sam, my wife and our children.
This has taken a long time, thank you all for your love, support and patience.

Contents

List of Tables

List of Figures

Abstract

Although many Mathematical Programming techniques have been developed for application to decision making under uncertainty, these techniques are based on three implicit assumptions. The first is that probabilities can be determined for the outcomes of the uncertain parameters, the second is that the decision maker is risk neutral, and the third is that all of the decision maker's concerns can be included in the formulation. While there are many decision making situations for which these assumptions are appropriate, there are many other situations for which they are not. In particular, these assumptions are seldom supportable for strategic decision making problems. Strategic decision making must consider possible future events that have seldom, if ever, occurred before, and for which probabilities cannot be determined. Because the situation will occur only once, and the decision will have a large impact, the decision maker is unlikely to be risk neutral. Finally, the decision makers will often have concerns that cannot be represented in a mathematical programming formulation.

In this work we present an approach to decision making under uncertainty that relaxes the three assumptions listed above. We assume that the uncertain future can be described as a small set of scenarios. These scenarios can be considered to have separate, competing objectives, because decisions that prepare well for one scenario generally prepare poorly for the others. The problem is formulated as a multiobjective optimisation problem, and a set of non-dominated decisions is found. The decision maker can choose a decision from this set according to his/her attitudes to risk, and to account for other requirements that cannot be represented in a mathematical programming formulation. This approach is developed for problems with continuous variables, and then extended to problems that include binary variables.

Chapter 1

Introduction

1.1 Introduction

In this study we present a new application of mathematical programming techniques to decision making under uncertainty. This approach is designed for the analysis of situations in which the uncertain future can be modelled as a set of scenarios, but it is not appropriate to assign probabilities to those scenarios. This makes our approach applicable to a wider range of problems than can be handled by stochastic optimisation, which assumes that the scenarios have been assigned probabilities. Rather than finding a single solution for the problem, the method presented here finds a set of non-dominated solutions for the decision maker to choose among. For this reason we have called the approach "Noninferior Set Scenario Analysis".

In this chapter we discuss some of the issues involved in strategic decision making, and why strategic decision making is significantly different from operational decision making. We consider different methods used to represent the uncertainty faced by strategic decision makers, and the ways in which the representations assumed for stochastic optimisation differ from the representations used by planners and futurists. We then discuss ways in which the decision process can be modelled, and consider reasons why decision makers may be unwilling to implement the answers recommended by a stochastic optimisation model. In the final section we use a classification of uncertainty (Courtney, Kirkland & Viguerie 1997) that helps identify where there is a mismatch between stochastic optimisation and strategic decision making. We then outline our work, which we believe is applicable to a more general class of strategic decision making problems than is the case for stochastic

optimisation.

Before proceeding further, it is necessary to discuss how the words *uncertainty* and *risk* will be used. The Concise Oxford Dictionary describes *uncertain* as: "**1** not certainly knowing or known **2** unreliable **3** changeable, erratic", and *uncertainty* as: "**1** the fact or condition of being uncertain **2** an uncertain matter or circumstance". *Risk* is described as: "a chance or possibility of danger, loss, injury, or adverse consequences". These definitions suggest that in general usage, uncertainty is a neutral term used to describe situations which cannot be known with surety, whereas risk emphasises the downside.

In the decision making literature the term *decision making under risk* refers to situations in which all of the possible outcomes can be listed, and these outcomes can be assigned probabilities. The term *decision making under uncertainty* refers to situations in which the second, or both, of these conditions are not met. However, 'decision making under uncertainty' is also used to denote both situations. In particular, the stochastic optimisation literature uses the term 'decision making under uncertainty', despite the fact that probabilities are assumed for the uncertain events.

In this thesis, *uncertainty* will be used as a general term, conforming to the definition given by the Concise Oxford Dictionary, and whether or not probabilities are available will be stated explicitly. When using the terms *risk averse*, *risk neutral* and *risk taker* to describe decision makers, we will be using *risk* as defined by the Concise Oxford Dictionary. So a 'risk averse' decision maker is one who places a high value on avoiding adverse consequences, or 'downside' risk. A risk averse decision maker is willing to sacrifice performance under desirable outcomes in order to guard against undesirable outcomes. A 'risk taking' decision maker, on the other hand, will accept poor performance under undesirable outcomes in order to obtain high performance if things turn out well. The 'risk neutral' decision maker is not influenced to a greater, or lesser, degree by the prospect of desirable, or undesirable, outcomes occurring. In an optimisation context, a risk neutral decision maker will optimise the expected value of the outcomes. That is, the sum of the outcomes weighted by their probabilities.

1.2 Characteristics of Strategic Decision Making

Strategic decision making problems have characteristics that distinguish them from operational decision making problems and make them less amenable to solution using optimisation techniques. These characteristics will be outlined here, and then discussed further in the following sections.

The most obvious difference between operational and strategic decision making is the length of the time horizon. For strategic decision making the horizon will be measured in years or decades, whereas operational decision making uses horizons of months, weeks, days, or even hours. Long horizons mean that many of the outcomes of taking the decision will not be observed for years, which means that these outcomes should be treated as externalities, because they will not impact on the decision makers, but on future generations. This problem is compounded by the use of discount rates, because long time frames reduce the NPV of many outcomes to very small values, which means that these outcomes will have little impact on the choice of decision.

Generally, strategic decisions are important decisions because they involve commitment of significant resources, and they are difficult to reverse. They are also one-off in nature because the same situation will not occur again, which means that a bad outcome on one occasion cannot be offset by a good outcome on another. These characteristics of importance and non-repeatability mean that strategic decision makers cannot be assumed to be risk neutral. Finally, because strategic decision making usually involves multiple decision makers, and many stakeholders, strategic decisions must frequently be taken, and implemented, in the face of conflict.

1.2.1 Long Time Horizons and Discounting

The principal characteristic that distinguishes between strategic and operational decision making is the need to consider long time horizons, (measured in years or decades), and this creates special problems. As decision makers look further and further into the future, they see increasing numbers of uncertainties and these uncertainties become more and more difficult to describe. There may be uncertainty about changes that will gradually become apparent over time, and there may be uncertainty about events that will occur at a particular point in time. Decision makers will be uncertain about when particular events will occur, or even whether they will

occur at all. They will also be uncertain about the set of possible outcomes, and about the probabilities that should be assigned to those outcomes. It is exceedingly difficult to determine a probability distribution for events that have seldom, if ever, occurred before, but it is just this type of uncertain event that strategic decision makers must consider. The situation is further complicated when decision makers are uncertain about gradual changes. Gradual changes are not usually associated with discrete events, and it is unusual for the change to be recognisable at a particular point in time. Never-the-less, changes that evolve gradually over time can be a very important part of a strategic planning problem, and they can involve high levels of uncertainty.

Long horizons also create difficulties when the problem includes a flow of costs and benefits, some of which will occur in the distant future. To evaluate alternative decisions, their flows of costs and benefits over time must be compared. If the costs and benefits can be expressed as cash flows, then comparisons are normally made by discounting all cash flows to a net present value. However, high discount rates devalue future cash flows to the point where they have very little impact on the net present value. For example, at a 10% discount rate, $1,000 in 10 years time has a net present value of $385, and $1,000 in 30 years time has a net present value of only $57. Future cash flows "disappear from sight", and this leads to a form of myopia in long term decision making. For decision makers who are focusing on the immediate decision this can be comforting, because it means that events in the more distant future can be ignored. However, this works against the intention of long-term planning, which is to take this distant future into account.

One reason for using high discount rates is to provide for uncertainty by reducing the value placed on uncertain positive cash flows that will occur in the future. However, a high discount rate also devalues negative cash flows, and so a high discount rate used to provide for downside risk will have the opposite effect when applied to future costs. If discount rates are used as a surrogate for modelling downside risk, then a case can be made for using high discount rates for positive cash flows, and low discount rates for negative cash flows.

Explicit modelling of the uncertainty reduces the need to use discount rates as a surrogate for modelling downside risk, but the problem remains of comparing cash flows that occur at different times. Over long horizons discount rates set close to current borrowing rates cause future cash flows to "disappear", and rates must be

set close to zero to avoid this difficulty. The discount rate could be set equal to the inflation rate, so that cash flows are in constant value dollars. However, this requires prediction of the inflation rate, which is also uncertain. In any event, decision makers may find it difficult to justify using very low discount rates for long term planning, when higher rates will be used for operational planning.

1.2.2 Externalities

When long planning horizons are used, some of the effects of a decision become externalities. The decision makers know that they, themselves, will not benefit, or suffer, from the effects of their decisions that will occur in the distant future. This produces a special form of externality, in which the effects of a decision do not only impact on other people, they also impact on other times. For example, the operation of a nuclear power station for the next 30 years will produce benefits for the decision makers in terms of electricity generated, but it also produces time externalities in the form of decommissioning costs, and the ongoing costs and dangers of storing radioactive waste products with half lives of hundreds, and even thousands, of years. Positive time externalities also occur. For example, trees that are planted to provide erosion control in the short term may eventually become saleable timber, and generate a positive time externality.

Time externalities contribute to the general problem of deciding how to compare costs and benefits that are widely separated in time. The effects of a strategic decision occur over long time horizons, they are both positive and negative, and an effect may be positive for some people and negative for others. In addition, these effects are often in different units, and there will be uncertainties about their magnitude, about their timing, and even about whether the effects will actually be beneficial or detrimental, and to whom. The problem of externalities is heightened by the use of discount rates, because they reduce the impact of future cash flows on the NPV of strategic decisions.

1.2.3 Strategic Decisions Are Important

Another distinguishing characteristic of strategic planning is the importance of the decisions. Operating decisions are short-term in nature, and it is very unusual for a

poor operating decision to threaten the survival of an organisation, although a sequence of poor operating decisions might do so. Operational decision making usually includes rapid feedback of results, and the opportunity to make adjustments at low cost. On the other hand, a strategic decision that goes wrong can lead to disaster. Strategic decisions are long term in nature, they commit significant resources, and it can be difficult and expensive to make adjustments later. An organisation may have to wait several years before a strategic decision can be evaluated, by which time it may be too late to correct matters. For example, consider a company that decides to build a new timber processing facility to replace its existing operation, and to increase its capacity. The construction goes completely to plan and is within budget, but, just as the facility is being commissioned, the price of timber collapses and the company goes into receivership. The company made a good decision, based on what they knew and could foresee at the time, feedback from the project showed that things were going well, but they were "ambushed" by an abrupt change in their operating environment.

1.2.4 Attitudes to Risk

This example also illustrates why decision makers are often risk averse when making strategic decisions. The penalties of getting it wrong can be severe, and it may be difficult, or impossible, to make corrections in time to save the situation. Strategic decisions are also one-off in nature, whereas operational decisions are usually repetitive. When a decision making problem recurs regularly, the decision maker can expect the long-run result of following a particular decision policy to converge on the expected value. However, when decisions are taken once only, and the situation will not occur again, only one of the possible outcomes can occur. Under these conditions the decision maker may not be willing to evaluate the candidate decisions on the basis of their performances averaged over all possible outcomes. Instead, a risk averse decision maker can be expected to choose a decision that avoids highly undesirable outcomes, even though that decision cannot produce an extremely good outcome either.

Of course, some strategic decision makers are risk takers, and will choose a decision because it is predicted to perform very well under some outcomes. Such decision makers are willing to accept the fact that their chosen decision will perform

very badly if less favourable outcomes eventuate.

1.2.5 Strategic Decision Making Involves Conflict

Strategic decision making frequently occurs in a context of conflict. Strategic decisions have the potential to affect many people, both positively and negatively, and these stakeholders will be in conflict over which objectives should be used to evaluate the decision. Differing views among the stakeholders also lead to conflict over the constraint set, and even whether some aspects of the problem should be expressed as objectives or as constraints. For example, should a low inflation rate and a low level of unemployment be objectives or constraints, or should they even be included? If they are expressed as constraints, what levels should they be constrained to? If they are both objectives, what are the trade-offs between them? This is further complicated by the fact that the stakeholders' objectives may change over time, and that the group of people who are stakeholders may also change over time. Consider a regional planning authority that is considering competing applications to dam a river for hydro generation, to divert the river for irrigation, or to maintain the river in its present state for recreation. Over the next twenty years the stakeholders will change, their objectives may change, and the criteria against which the state of the river is evaluated may also change.

The differing views of the stakeholders also leads to disagreement about the data. Different stakeholders can bring differing definitions of an economic indicator to the problem, and statistics can be manipulated to reinforce or undermine particular viewpoints. Stakeholders also make predictions about the future to support their points of view. There will be conflict over what will happen, what could happen, and even the extent to which the future should be considered at all. For instance, in the river planning example above, the power company could emphasise the danger of power shortages, the irrigation lobby the greater economic benefits of agriculture over electricity, and the kayakers the economic benefits of tourism. All three groups can be expected to speak on behalf of future generations (future stakeholders) and to make predictions about the future to support their cases.

Finally, there may be conflicts of interest between the personal goals of decision makers, and the goals of the organisation for which they are taking decisions. For example, a decision maker may be a manager on a three year contract whose annual

bonus depends on short-term company performance. This decision maker will experience a conflict of interest if some alternatives lead to a reduction in short-term performance and an improvement in long-term performance, while other alternatives enhance short-term performance at the expense of long-term performance. Similarly, in the river planning example, there would be a conflict of interest for an engineer who would have interesting work for years if one of the development options is chosen, but can expect to be made redundant if the river is left in its present state.

A related issue is that strategic decisions frequently lead to the redistribution of costs and benefits. For example, a decision to widen a main road as part of initiatives to relieve traffic congestion may benefit people who commute to work from an outlying suburb. However, the people who live along the road will lose their front gardens, and will have to live with increased noise and pollution. Such allocations of costs and benefits to different groups of people should be seen as an important issue for strategic decision makers, and they can certainly be expected to lead to conflict.

1.3 Modelling the Uncertainty

1.3.1 Forecasts and Scenarios

An extreme response to uncertainty is to attempt to remove it from the problem by producing a forecast. The forecaster attempts to resolve the uncertainties into a forecast of what is going to happen. Although a forecast may be qualified by statements of potential exceptions, and future parameter values may be expressed as intervals rather than exact point values, the intention is to provide a deterministic view of the future for the decision maker to plan to. Decision making based on forecasts works reasonably well when the environment is stable and the future will unfold in much the same way as the recent past. However, when the future turns out to be very different from the past, decisions based on forecasts may perform very poorly, and can lead to disaster. The shortcomings of using forecasts as a basis for planning have been recognised for many years, and a great deal of work has gone into developing better approaches (see, for example, Wack 1985*a*, Wack 1985*b*).

In the late 1960's, planners at "Royal Dutch/Shell" adapted the scenario approach of Herman Kahn for use in the corporate planning environment. By the

early 1970's Shell's planners had developed their "scenario planning" technique and had it accepted by Shell management (Wack 1985*a*, Wack 1985*b*). The use of scenario planning enabled Shell to prepare for the oil industry's switch from a buyers' market to a sellers' market which precipitated the oil shock of 1973.

Shell's planning group had embarked on experimental studies to explore the business environment of the year 2000. One of these studies revealed that the then current environment, in which the oil producing countries were willing to make unlimited quantities of oil available at low prices, could not continue. This insight lead the planners to realise that the oil industry must inevitably suffer a major disruption.

The planners developed contrasting scenarios in which they described how the future might evolve. One group of scenarios described the future implied by the impending disruption, whereas another scenario (Wack called it the "three-miracles scenario") described the events required to maintain the current high demand/low price environment. The planners used these scenarios to convince management that change was inevitable, and that Shell should prepare for that change.

This example illustrates one way in which scenarios are used. The planning group concluded that an important change was inevitable, although its timing was uncertain. However, they realised that if they were to forecast the event, the forecast would be rejected out of hand by management because it contradicted management's world view. So, instead of forecasting the event, the planners developed contrasting scenarios to describe ways in which the future might evolve. When they were presented with the scenarios Shell's management realised that the scenarios that were consistent with their world view of "continuing expansion" were implausible. They then changed their world view, and took on board the inevitability of change.

More usually, scenarios are developed to describe different paths into the future, all of which are thought to be possible (although some of them may be considered unlikely). Scenarios of this type may be qualitative "stories" about how events may evolve, or they may be quantitative models, suitable for use in mathematical programming, or other quantitative analysis. In the planning literature opinion is sharply divided over the question of whether scenarios should be qualitative or quantitative. See Schnaars (1987) for a discussion of this debate. Opinion is less divided on the question of how many scenarios should be prepared. Schnaars (1987) reports:

> There seems to be a consensus in the literature that three scenarios are best. Some schemes propose only two, and some propose more than three, but the general feeling is that two tend to be classified as 'good-and-bad', while more than three become unmanageable in the hands of users, resulting in their attending to only a subset anyway.

However, there are dissenting opinions. In particular, there is concern that when three scenarios are used, attention will focus on the scenario that appears to represent the "middle ground", even to the extent of treating it as a forecast. Mobasheri, Orren & Sioshansi (1989) report that at Southern California Edison they selected twelve quantitative scenarios (from an initial set of 45) to describe how the future might evolve.

The Management Science and Operations Research literatures assume quantitative scenarios (typically without even considering the possibility that scenarios might be qualitative), and large numbers of scenarios are common. The Russell-Yasuda Kasai Asset/Liability Model (Carino, Kent, Myers, Stacy, Sylvanus, Turner, Watanabe & Ziemba 1994), works with 256 scenarios, and techniques such as *importance sampling* (Dantzig & Infanger 1993, Infanger 1994) are designed for problems with thousands of scenarios.

The real difference, however, is not in the number of scenarios, but in what is meant by a scenario. In the planning literature, a scenario is described as:

> ... a complete, consistent, and plausible description for a possible future state of the world that could occur if one or more major events were to happen.
>
> (Mobasheri et al. 1989)

And Wack (1985*a*) describes *decision scenarios* as:

> They create a few alternative and internally consistent pathways into the future. They are not a group of quasi-forecasts, one of which may be right. Decision scenarios describe different worlds, not just different outcomes in the same world. Never more than four (or it becomes unmanageable for most decision makers), the ideal number is one plus two; that is, first the surprise-free view (showing explicitly where and why it

> is fragile) and then two other worlds or different ways of seeing the world that focus on the critical uncertainties.

In other words, the planning literature uses the term scenario to mean "a story about the future". Scenarios are intended to highlight the ways in which the future may differ from the present and the past, but they are not predictions. To quote Wack again:

> The point, to repeat, is not so much to have one scenario that "gets it right" as to have a set of scenarios that illuminates the major forces driving the system, their interrelationships, and the critical uncertainties.

In contrast, the stochastic optimisation literature uses the term scenario to describe a combination of events constructed by taking one outcome from each of a set of independent random variables that have been chosen to describe the uncertain parameters. If the random variables are discrete, then the scenarios are created by taking every possible combination of the outcomes of the uncertain events. Typically, the scenarios are never explicitly constructed, or described, but the solution algorithms generate the scenarios (or a subset of them) as required. There is the implication that the scenarios describe "real futures" that could actually occur, and it is common practice to assign them probabilities. Because the probabilities sum to one there is the implied assumption that the scenarios include all possible futures, that is, that no other future can occur. While it is unlikely that the modellers actually believe this to be the case, the assumption is implicit in the modelling, and may be lost sight of when interpreting the results. This use of probabilities is in stark contrast to the frequently, and strongly, expressed view in the planning literature, that:

> Another criticism, aimed particularly at the quantitative methods, is that it is probably folly to attempt to assign probabilities to scenarios. Such estimates can be nothing short of misleading. The precision they imply is not warranted by either (1) the data that was used to derive them, or (2) the phenomenon they purport to predict. Scenarios are possibilities, not probabilities.
>
> (Schnaars 1987)

Probability distributions may be calculated directly from historical data, or they may be derived from beliefs and assumptions about the system that will normally be based on previous experience. If random variables for future events are calculated from historical data, or based on previous experience, then the analyst must be assuming that the future will be similar to the past. Although this assumption is supportable for operational uncertainties, strategic planning problems include uncertainties that are new, and for which there is little historical data.

1.3.2 Cross-Impact Analysis

Experts can be called upon for suggestions of possible future events (and even for predictions) to be used to construct scenarios. The experts may also be asked to provide probability estimates for the events, which will be used to determine probabilities for the scenarios. In cross-impact analysis the experts are asked for estimates of the probabilities that particular events will occur, and for estimates of the conditional probabilities of the events, when the events are taken in pairs. These estimates are then processed by a computer simulation, or mathematical programming model, to produce a single 'most likely' scenario, or multiple scenarios ranked by probability. For example, the mathematical programming model, SMIC 74 (J.C. & Godet 1975) uses quadratic optimisation to provide probability estimates for every possible scenario, while Kluyver & Moskowitz (1984) use interactive goal programming to achieve the same end. Helmer (1981) proposes an interactive simulation approach to investigate the interactions between the random events and the interventions available to the decision makers.

Cross-impact analysis has been severely criticised on several fronts. Some of the methodologies employed generate probability values that are inconsistent with the basic rules of probability, for example, they may be outside the range 0 to 1, and conditional probabilities may not sum to 1, (see Helmer 1981). There is considerable argument about what the conditional probabilities mean, how they should be interpreted, and whether events are correlated, or have causal relationships (Enzer & Alter 1978). However, the fundamental question about cross-impact analysis is whether the judgmental estimates of experts are suitable input data for such complex manipulation, and whether the derivation of such precise and detailed results are justified.

> The key problem with Cross-impact is that judgmental estimates are surely not amenable to any such mathematical machinations. As Kelly (1981) notes, 'to suggest that any method exists which might extract such blood from such stones is wishful thinking' (p. 342). McLean (1981) adds, 'in putting the emphasis on computation rather than conceptualization it tended to conceal the contradictions inherent in the approach' (p. 349)
>
> Schnaars (1987)

1.4 Modelling the Decision Process

Strategic decision making is an ongoing process in which decisions are implemented, and plans are formulated and subsequently revised. Decision makers combine their understanding of the current situation with estimates of how the future may turn out, and apply a set of objectives to decide on immediate decisions and to formulate plans for the future. After some period of time has elapsed, the decisions already implemented will have interacted with events to create a new situation, and the decision makers must formulate new decisions and plans. It is important to distinguish between decisions and plans. A decision is a commitment to act, whereas a plan is a statement of intent, normally conditional on the evolution of events. Rosenhead (1989, page 198) distinguishes between decisions and plans as:

> A decision is a commitment of resources which transforms some aspect of the decision maker's environment; the environment can be restored to its former condition, if at all, only by a further decision and a (at least) psychological cost. A plan consists of a foreshadowing of a set of decisions which it is currently anticipated will be taken at some time or times in the future, or an identification of an intended future state which necessarily implies such a set of future decisions.

It is important to recognise this distinction when formulating and interpreting models used to support decision making under uncertainty. In the stochastic optimisation literature, this distinction is blurred by the use of the terms *immediate decision* and *recourse decision*. The immediate (or stage one) decision really is a decision. It will

be taken immediately, and it commits the organisation to some course of action. Recourse decisions, however, are plans, conditional on the implementation of some immediate decision, and on the outcome of the uncertainty. However, a recourse decision may actually be the implementation of an immediate decision. For example, consider a stochastic optimisation model, in which one branch of the event tree models failure of a power station, and the recourse decision is to buy in power from a neighbouring utility. If the recourse decision models the contractual statement: "If our plant breaks down we undertake to meet the full demand for power by buying from a third party at whatever price we have to pay.", then the recourse decision is not a plan, it models the impact of a decision to guarantee supply. However, if it is included as a means of pricing the uncertain event of plant failure, and the company considers itself free to not meet the full demand, then the recourse decision is a plan which will be reviewed if the failure actually occurs.

It is also important to recognise that the recourse decisions in the model are limited to those realisations of the future, and those alternatives, that were included in the model. Normally, the events that occur will not match any of the futures in the model, and the decision maker must analyse the new situation before implementing a decision. Even if events follow a branch of the event tree very closely, the decision maker should still analyse the situation afresh, taking advantage of any new information that is now available. In any event, the recourse decision (plan) will now be the immediate decision, the future will still be uncertain, and so there will be a new planning problem to be formulated and solved.

Not only should recourse decisions be seen as representing plans, they should also be seen as representing the implications of the immediate decision. Some of the recourse decisions for a particular choice of the immediate decision represent actions that the organisation expects to have to take if events fall out in a particular way. When viewed in this way, such a recourse decision is not so much what the organisation plans to do, but what the organisation expects to be forced to do. For example, it is probably more appropriate to view a recourse decision that fires 20% of the production staff as an action the company would be forced to do, rather than one it plans to do. However, an immediate decision may also have positive implications. For example, under another scenario, the recourse decision for the same immediate decision may be to hire additional staff.

1.5 The Decision Maker's Perspective

President Truman had a sign on his desk that said "the buck stops here", and this sign also belongs on the desks of decision makers. Unlike the support staff who create scenarios, build models, and carry out the analysis, being aware of what might happen is not, in itself, sufficient. Decision makers must actually decide what to do, now; and they must answer for the consequences. The other important difference between decision makers and their support staff, is that the decision makers will (should?) be concerned about the whole problem, and all of the goals, objectives, and constraints that make it up, whereas the support staff need not be. There are many reasons why aspects of the problem will not be available to the support staff. There may be a desire to maintain secrecy, the decision makers may still be formulating the problem, or the decision makers may be reluctant to articulate goals, constraints and preferences explicitly. Generally, however, this is a result of cognitive overload. No one can know everything about everything, decision makers will know about the whole problem in a general way, whereas the support staff will have detailed information about a few parts of the problem, and know very little about the rest of it. This reluctance and/or inability to communicate the whole problem to the analyst means that decision makers cannot expect to be given the optimal solution to their problem, and that the analyst should not attempt to do so. What decision makers should be looking for is insights and information to facilitate their search for an implementable decision.

> The ostensible purpose of a mathematical programming model is to optimise a stipulated objective function subject to stipulated constraints. But its *true* purpose, at least in strategic applications, is to help develop insights into system behaviour which in turn can be used to guide the development of effective plans and decisions.
>
> (Geoffrion 1976)

Decision makers bring their own perceptions and expectations to a problem situation, and they can be expected to adjust, or even reject, modelling results that do not match their world view. If the output from a modelling exercise is a single optimal solution that does not fit the decision makers' expectations then the decision makers must adjust their own expectations, adjust the solution, or tell the

analyst to find a better one. Decision makers are very unlikely to adjust their own expectations in response to a single undesirable answer. They are much more likely to adjust the model solution to match their world view, or to reject it out of hand and select a decision by other means.

The representation of the problem can now be viewed as consisting of two parts, the first being the objectives and constraints represented in the analyst's model, and the second being the objectives and constraints implied by the decision makers' perceptions and expectations. If the decision makers decide to adjust the solution provided by the model, then that solution must be considered to be inadequate with respect to the second part of the problem. However, if the decision makers adjust the solution without consultation with the analyst, then the decision finally implemented may be inadequate with respect to the modelled part of the problem. If the decision makers ask for another solution, and give the analyst additional criteria, then a satisfactory solution should be found, especially if they carry out several iterations. However, the decision makers may be unwilling, or unable, to go back to the analyst because of secrecy or time constraints. In such situations, the analyst would best serve the decision makers by providing a set of contrasting alternatives for them to choose among, and information about how sensitive these alternatives are to changes in the data. What the decision makers need are insights of the form: "The optimal solution consists of these actions and produces this objective function value (or values if the model is multiobjective), but there are also these other sets of actions that lead to these objective function values." They also need to be told about the sensitivity of the alternative courses of action to changes in the data. They need additional insights of the form: "Alternative A produces these objective function values under the assumed data, however if the data is changed in this manner, then Alternative A will produce these objective function values." The decision makers can now make an informed choice between alternatives. They can balance the cost to the modelled parts of the problem of choosing actions that fit the parts of the problem that weren't modelled. If there is time for another modelling round, then the decision makers and the analyst will have more information with which to design the next round than would be available if only a single, optimal solution had been furnished the first time.

Strategic decisions are implemented in an environment of continuing change and uncertainty. At best, only some of the events that were foreseen and considered when

the decision was chosen will actually occur. In addition, new potential threats and opportunities will appear, some of which may occur and many will not. Not only is it impossible to predict the future, it is impossible to foresee all of the alternatives. In the face of this high level of uncertainty decision makers must look for decisions that at least keep their options open, but preferably enhance their ability to respond to events. Not only do they need to be able to protect against threats, they should be able to exploit opportunities. In light of these observations, it is apparent that the optimal solution to any mathematical model of the situation will soon be outdated, and the decision makers will be addressing the strategic planning problem again. The decision makers' evaluation of the decisions already taken should not be "were they right", but "how well did they prepare the organisation for the situation it now finds itself in" and "did they enhance or inhibit the organisation's ability to respond to the threats and opportunities that actually occurred?".

1.6 Outline of this Study

In the context of strategic decision making, Courtney, Kirkland & Viguerie (1997) suggest that uncertainty comes in four levels, and that decision making becomes increasingly difficult as problems move from the first level to the fourth level. At level 1, the future is sufficiently predictable to be described by a single forecast. While the future may be uncertain, the uncertainties are irrelevant to the choice of strategic decision. At level 2, the uncertainties are significant, and they can be summarised into a small number of alternatives, or scenarios. It may also be possible to assign probabilities to the scenarios. At level 3 the uncertainties describe a range of possible futures, but this range does not naturally summarise into scenarios. It may be possible to develop a set of representative scenarios, but these scenarios will not describe all of the possible paths into the future. It will certainly not be possible to assign probabilities to them. Finally, there is level 4, in which 'anything could happen'. It is no longer possible even to determine ranges for the outcomes, and it is certainly impossible to create plausible scenarios.

The categorisation of strategic planning problems into these four levels provides a useful framework within which to consider the use of scenarios to model future uncertainty. It seems clear that authors such as Wack and Schnaars, writing in the planning and forecasting literature, are assuming level 4, or difficult level 3

situations, in which it is impossible to completely describe the future, and it is also difficult, or impossible, to derive quantitative scenarios. They seem to implicitly assume that all decision making under uncertainty is at level 3 or 4, and their criticisms of approaches that derive quantitative scenarios appear to be based on that assumption. However, Mobasheri et al. (1989) do develop quantitative scenarios, and they would appear to be working in an environment that Courtney et al. (1997) would describe as level 3, or perhaps, level 2.

Clearly, the stochastic optimisation literature is assuming level 2 problems, in which scenarios are readily apparent, and for which probabilities can be determined. There are examples in the stochastic optimisation literature in which stochastic optimisation has been applied to level 3 and level 4 problems, for example to the issue of global warming and CO_2 emission policies (e.g. Birge & Rosa 1995). For problems such as this the criticisms made in the planning literature would seem to be most appropriate. Global warming is clearly a level 4 (or perhaps a difficult level 3) problem. While it is probably useful to model the future as a small set of representative scenarios, to assign probabilities to these scenarios would seem to be quite inappropriate. To then formulate the problem as a stochastic optimisation problem that optimises the expected value of an objective function over these scenarios, and to derive insights from the resulting single point solution is quite self deceiving.

We suggest that mathematical programming can make a useful contribution to strategic decision making provided that the uncertainty is at level 1, or level 2. If the assumption that scenario probabilities are available can be relaxed, then we believe that mathematical programming could be applied to strategic planning problems in which the uncertainty is at level 3.

The purpose of this study has been to develop a mathematical programming approach to decision making under uncertainty that addresses some of the shortcomings of stochastic optimisation, so that it can be applied to level 3 strategic planning problems. The intention is to make mathematical programming more applicable to problems in which the uncertainty cannot be described reliably, and in particular to strategic planning problems to which stochastic optimisation is not well suited. As has been discussed in the previous sections, there is a mismatch between the representations of future uncertainties being developed by planners and futurists, and the types of problem for which stochastic optimisation has been designed. In particular, stochastic optimisation implicitly assumes that the scenarios

represent actual futures that can be assigned probabilities, and that the decision makers are risk neutral and wish to optimise an expected value over the scenarios. In contrast, many authors in the planning literature recommend that scenarios be representations of contrasting "possibilities", and that no attempt should be made to create scenarios that are alternative forecasts. These authors also express the view that the scenarios should not be assigned probabilities.

A fundamental assumption that is implicit in mathematical programming techniques that are designed to produce a single, optimal solution, is that all of the concerns of the decision makers can be represented in the model. This assumption is applied by the description of the solution as "optimal". In many decision making situations this assumption reasonable, and the decision maker will be willing to implement the decision produced by the model. However, as discussed in Section 1.5, there are many situations where this is not the case, and this is particularly true of strategic decision making problems. Because the approach presented here produces multiple decisions for the decision makers to choose among, it is able to relax the assumption that the whole of the problem can be represented in the model. The decision makers can apply the aspects of the problem that could not be included in the model to the process of choosing among the alternatives identified by the analysis.

Stochastic Optimisation also requires that the objective function be in the same units under all scenarios, so that a single, composite objective function can be formed for the problem. However, as pointed out in Section 1.2.5, the stakeholders' objectives may change over time, and this may well imply a change in units. The change could be represented by scenarios with different objectives expressed in different units. In the example of the regional authority, one scenario could have an objective of maximising the number of days the river flows are high enough for rafting and kayaking, while another might have the objective of maximising the revenue produced by power generation. For this problem to be formulated as a stochastic optimisation problem, the two objective functions would have to be brought to the same units. This could be done by assigning a monetary value to days in which flows are high enough for rafting and kayaking, but the derivation of such values is extremely difficult. The work presented here avoids this problem. The scenarios are viewed as having independent objectives, and there is no need to combine these objectives into a single objective function.

Our work is intended to extend the applicability of mathematical programming techniques to level 3 problems. Such problems can only be described using a set of representative scenarios, and these are the type of problem discussed by many authors writing in the planning literature (e.g. Wack 1985*a*, Schnaars 1987). These authors recommend that two or three, but never more than four, scenarios should be prepared, and so we have limited the main part of our study to problems with two or three scenarios. However, in our mixed integer example we have used five scenarios, and it is interesting to see how the complexity and computational effort involved in analysing the problem increased significantly, but that little more information was produced than could have been obtained using three scenarios.

We adapt a set generation technique from multiobjective optimisation to the problem of decision making under uncertainty. This enables us to develop an approach that relaxes the assumption that the scenarios represent "real" futures and can be assigned probabilities. The main body of this work develops the approach for problems with continuous variables. However, strategic planning problems frequently include binary variables and we have extended these ideas to the mixed integer problem.

In Chapter 1 we have outlined some of the issues involved in strategic decision making, and why strategic decision making is significantly different from operational decision making. We then discussed different methods used to represent the uncertainty faced by strategic decision makers, and the ways in which the representations assumed for stochastic optimisation differ from the representations used by planners and futurists. Finally, we discussed ways in which the decision process can be modelled, and we considered reasons why decision makers may be unwilling to implement the answer recommended by a stochastic optimisation model.

Chapter 2 is our literature review. First we briefly describe the development of stochastic optimisation techniques over the last forty years. Next we consider some other approaches to modelling uncertainty. These are probabilistic programming, stochastic dynamic programming, fuzzy programming, and the related interval and grey programming. We then discuss recent extensions to stochastic optimisation, and reasons why stochastic optimisation has limited applicability to strategic decision making. In Chapter 3 we review those sections of the multiobjective optimisation literature that are relevant to the work reported in this thesis. That is, set generation techniques, and the implications of including integer variables in a

multiobjective optimisation problem.

In Chapters 4 and 5 we develop the theory behind our approach, which we call "Noninferior Set Scenario Analysis". In this approach the problem is formulated so that each scenario has its own objective function. It can then be analysed as a multiobjective optimisation problem in which the selection of the stage one decision involves trading off between the competing objectives of the scenarios. In our approach we find an approximation to the set of noninferior stage one decisions. The decision maker can then choose a decision to implement from this set of alternatives. In this context, a stage one decision is noninferior if it is not dominated by another stage one decision. Stage one decision A dominates stage one decision B if at least one of the scenario objective function values of decision A is better than the corresponding value of decision B, and none are worse. Because we treat the scenarios as having their own objective functions, and we find a set of noninferior decisions to choose among, it is not necessary to assign probabilities to the scenarios. However, once the noninferior set has been found it can be used to provide the optimal expected value solution for any set of scenario probabilities, and to provide sensitivity analysis on the probabilities. This work considers the two-stage stochastic problem only, and we have not attempted to generalise it to the multistage problem.

This formulation, in which each scenario is viewed as having its own objective, means that trade-offs between the objectives have a different interpretation than is the case in multiobjective optimisation. In multiobjective optimisation, for any chosen decision every objective will have a value. This means that poor values of some objectives are compensated for by good values in others. In contrast, when the objectives are scenario objectives, only one objective value will be observed, because only one scenario will occur. Another way of viewing this distinction is that the decision maker trades off the possibility of a good result under one scenario against the possibility of a poor result under another. In the event, the decision maker will only get one of the results. This comment applies equally to the stochastic optimisation formulation. Only one scenario will occur, and the decision maker will only observe the outcome under that scenario. This is obscured when the expected value of the solution is reported. It may appear that the decision maker will obtain the expected value, whereas the value of the solution will actually be the objective function value of the scenario that does, in fact, occur.

The solution technique used in Noninferior Set Scenario Analysis, or NSSA, is

based on the work of Cohon, Church & Sheer (1979) who developed an algorithm for finding an approximation to the noninferior set for bi-criterion problems, and Solanki, Appino & Cohon (1993) who extended the approach to problems with more than two objectives. In Chapter 4 we adapt the bi-criterion algorithm to handle problems with two scenarios, and in Chapter 5, we apply the approach of Solanki et al. (1993) to problems with three scenarios. Finally, we consider the problem of presenting multiobjective optimisation results to decision makers, and we present some methods for doing this.

In Chapter 6 we take two examples from the stochastic optimisation literature and solve them using NSSA. The results obtained are compared to the results obtained using stochastic optimisation techniques. For the first example NSSA requires more computational effort than stochastic optimisation, but it also produces considerably more information. In the second example, NSSA produces more information than stochastic optimisation, and does so with less computational effort.

In Chapter 7 an algorithm is developed to analyse strategic decision making problems when they include integer variables. Frequently, strategic planning problems include alternative courses of action that must be taken in their entirety, or not at all. Because such problems cannot be modelled adequately using continuous variables, binary variables must be used. We have assumed a rather simplified problem structure in which all of the stage one variables are binary, and the recourse variables are continuous. This structure makes the resulting problem easier to solve than it would be if stage one and/or stage two had mixed integer decision variables. Although this assumption is rather restrictive, it does approximate many strategic decision making situations in which the stage one problem is to choose between alternative courses of action, and the stage two problem is to operate in the resulting environment.

Chapter 8, applies the algorithm to a hypothetical capacity expansion problem from the electricity industry. In this problem a company must start building generating capacity now, in order to enter an electricity market in ten years' time. The company has developed five scenarios to describe how this electricity market might be configured in the future, and we use our approach to identify nondominated alternatives for the company to choose among.

Finally, in Chapter 9 we summarise our work, draw some conclusions and consider avenues for future work.

Chapter 2

Optimisation for Decision Making Under Uncertainty: A Literature Review

2.1 Introduction

In this literature review we first consider the literature pertaining to stochastic optimisation as proposed by Dantzig and Beale, and the developments since then. We then consider other methods of modelling uncertainty, and extend the discussion to include alternatives to, and extensions of, stochastic optimisation.

In Section 2.2 we review the stochastic optimisation literature, and the development of techniques designed to cope with the computational intractability of large stochastic optimisation problems. In Section 2.3 we consider scenario approaches to modelling uncertainty, and in Section 2.4 we review chance constrained programming and its extensions and application in the forestry sector. Section 2.5 looks at stochastic dynamic programming, and its extension as dual dynamic programming, and Section 2.6 considers fuzzy, grey and interval programming, and applications of these techniques to decision making under uncertainty. We then consider extensions to, and implementations of, stochastic optimisation in Section 2.7. In particular, importance sampling, scenario aggregation and robust optimisation are discussed.

Finally, in Section 2.8 we summarise this review and conclude that the current applications of mathematical programming to decision making under uncertainty are not suitable for problems that Courtney et al. (1997) classify as level 3. Because

strategic decision making problems are typically level 3 (or level 4), this means that there is a significant gap in the mathematical programming techniques available to support strategic decision making. The work reported in this study is designed to help fill that gap.

2.2 Stochastic Optimisation

2.2.1 Historical Summary

Stochastic programming with recourse was first proposed independently by Dantzig and Beale in 1955, however these ideas were not vigorously pursued until the last two decades. Considerable theoretical work was done, but application of most solution techniques to real problems has been impractical, or even impossible, because of the lack of adequate computational capacity. Many algorithms were tailored to solve specific problems with special structures that could be exploited to minimise the computational effort required. Unfortunately, most real problems cannot be forced into these special structures.

During the last twenty years or so, the continuing search for solution techniques has been complemented by the rapid increase in computer power. This has made it possible to develop algorithms and computer codes that can handle real stochastic problems. These developments have been spurred on by electricity generating organisations which need to solve large, complex, and highly stochastic multi-period scheduling problems (e.g. Pereira 1989). Similarly, people working in the finance industries (e.g. Carino et al. 1994), and the forestry sector (e.g. Gassmann 1989, Hof, Kent & Pickens 1992), have developed and implemented large scale stochastic optimisation algorithms.

In 1969 the L-shaped algorithm was developed for solving two-stage stochastic linear problems with recourse (Van Slyke & Wets 1969), and this technique has been used as the basis for many algorithms designed to solve more complex stochastic problems. The L-shaped algorithm is an outer linearization procedure as in Bender's decomposition. Birge & Louveaux (1988) modified this method to apply several cuts to the problem at each iteration, and Birge (1985) extended it to the multi-stage case. However, Birge's implementation is restricted to problems with no more than three time periods and a small number of independent scenarios. Gassmann

(1990), wrote the computer code MSLiP, which he claims can handle any number of periods. Pereira & Pinto (1985) discuss a computational scheme for multi-stage power generation scheduling, also based on Bender's decomposition.

Rockafellar & Wets (1991) developed a method for aggregating scenario solutions into an overall solution to the stochastic optimisation problem. This technique has been applied by Robinson (1991) to portfolio optimisation, Jörnsten (1992) to the sequencing of oil and gas fields, and by Jönsson, Jörnsten & Silver (1993) to inventory problems.

The use of decomposition techniques to handle large problems leads naturally to the exploitation of parallel processing (e.g Rosen & Maier 1990), which has the potential to dramatically increase the size of problems that can be solved.

2.2.2 The Two-stage Stochastic Problem

A type of stochastic problem which has received considerable attention is the two-stage linear program with recourse. In its most general form it is:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimise}} \quad E_{\tilde{\xi}}\left\{\mathbf{cx} + Q(\mathbf{x}, \tilde{\xi})\right\} \\ &\text{subject to} \quad A\mathbf{x} = \mathbf{b} \\ &\qquad\qquad \mathbf{x} \geq 0 \end{aligned} \tag{2.1}$$

where: $Q(\mathbf{x}, \tilde{\xi}) = \min \{\mathbf{q}(\xi)\mathbf{y}(\xi) \,|\, W(\xi)\mathbf{y}(\xi) = \mathbf{h}(\xi) - T(\xi)\mathbf{x},\ \mathbf{y}(\xi) \geq 0\}$

the $m_0 \times n_0$ matrix A, and the vectors $\mathbf{c}$ and $\mathbf{b}$ are known

ξ is a random vector.

For each ξ , we have the $m_1 \times n_1$ matrices $T(\xi)$, and $W(\xi)$, and vectors $\mathbf{q}(\xi)$ and $\mathbf{h}(\xi)$. The technology matrix, $T(\xi)$, is a matrix of the effects of the decision values, $\mathbf{x}$, on the second stage. $W(\xi)$, the recourse matrix, is the matrix of coefficients of the second stage (or recourse) decision vector, $\mathbf{y}(\xi)$, in the stage two constraints, and $\mathbf{h}(\xi)$ is the right hand side of these constraints. $\mathbf{q}(\xi)$ is the objective function coefficients for $\mathbf{y}(\xi)$. Transposes have been eliminated for simplicity. $E_{\tilde{\xi}}$ represents the mathematical expectation with respect to $\tilde{\xi}$.

For a particular realisation of ξ, the problem reduces to the deterministic scenario

problem:

$$\begin{aligned} \text{minimise} \quad & z = \mathbf{cx}(\xi) \quad + \mathbf{q}(\xi)\mathbf{y}(\xi) \\ \text{subject to} \quad & A\mathbf{x}(\xi) \qquad\qquad\quad = \mathbf{b} \\ & T(\xi)\mathbf{x}(\xi) + W(\xi)\mathbf{y}(\xi) = \mathbf{h}(\xi) \\ & \mathbf{x}(\xi) \geq 0, \quad \mathbf{y}(\xi) \geq 0 \end{aligned} \tag{2.2}$$

where: $\mathbf{x}(\xi)$ = the optimal stage one decision for scenario ξ

$\mathbf{y}(\xi)$ = the optimal stage two decision for scenario ξ

There are various special cases of (2.1), with special structures that have been exploited to make them easier to solve. Principal among these simplifications are problems which are said to have *fixed recourse*, that is, the recourse matrix is deterministic ($W(\xi)$ becomes W).

When the second stage problem:

$$Q(\mathbf{x}, \tilde{\xi}) = \min \{\mathbf{q}(\xi)\mathbf{y}(\xi) \,|\, W(\xi)\mathbf{y}(\xi) = \mathbf{h}(\xi) - T(\xi)\mathbf{x},\ \mathbf{y}(\xi) \geq 0\} \tag{2.3}$$

is feasible for all outcomes of ξ, and all values of $\mathbf{x}$, the problem is said to have *complete recourse*. When stage two is feasible for all $\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq 0$ (i.e. for all feasible stage one decisions), then the problem is said to have *relatively complete recourse*. Complete recourse means that the stage two constraints are feasible under every scenario for all possible values of the stage one decision vector. Relatively complete recourse means that the stage two constraints are feasible under every scenario for those values of the stage one decision vector that are feasible in the stage one constraints. If a problem can be formulated to have complete, or relatively complete, recourse then the solution method does not have to ensure that stage one decision vectors are feasible in the second stage, and this greatly reduces the computational effort required to solve the problem.

When the problem has complete recourse, and the recourse matrix is deterministic, the problem has *complete fixed recourse*. That is, the constraint coefficients of the recourse vector are the same for all scenarios.

A special case of fixed recourse is *simple recourse*, where, with the identity matrix I of order m_1: $W = (I, -I)$. The second stage then reads as

$$Q(\mathbf{x}, \tilde{\xi}) = \min \{(\mathbf{q}^+\mathbf{y}^+ + \mathbf{q}^-\mathbf{y}^-) \,|\, \mathbf{y}^+ - \mathbf{y}^- = \mathbf{h}(\xi) - T(\xi)\mathbf{x},\ \mathbf{y}^+ \geq 0,\ \mathbf{y}^- \geq 0\} \tag{2.4}$$

An algorithm for solving stochastic programs with simple recourse, but with the additional simplification that the matrix, T, is also fixed, was developed by Wets (1983).

When the random variable, $\tilde{\xi}$ is discrete, the two-stage stochastic problem can be formulated as the *deterministic equivalent program* (DEP):

$$\begin{aligned} \text{minimise} \quad & z = \mathbf{cx} \quad + \sum_{\tilde{\xi}} p(\xi)\mathbf{q}(\xi)\mathbf{y}(\xi) \\ \text{subject to} \quad & A\mathbf{x} \qquad\qquad\qquad\quad = \mathbf{b} \\ & T(\xi)\mathbf{x} + W(\xi)\mathbf{y}(\xi) \quad = \mathbf{h}(\xi) \quad \forall \quad \xi \in \tilde{\xi} \\ & \mathbf{x} \geq 0, \quad \mathbf{y}(\xi) \geq 0 \end{aligned} \tag{2.5}$$

Stage two feasibility is enforced for all scenarios (all outcomes of $\tilde{\xi}$), by including all constraints for all scenarios in the problem. For large problems with many scenarios, the DEP is very large, and it may not be possible to solve it directly. This was certainly the case five or ten years ago before the recent, and dramatic, expansion in cheap computer power. The L-shaped algorithm (Van Slyke & Wets 1969) uses an outer linearization, as in Benders' decomposition, to break the problem down into manageable portions, and this algorithm forms the basis of many techniques for solving the multi-stage stochastic problem.

The L-shaped algorithm has been described in detail by many authors (e.g. Birge & Louveaux 1988, Dantzig & Glynn 1990, Kall & Wallace 1994), so it will only be summarised here. It starts by solving the first stage problem myopically without regard for the second stage. Each outcome of $\tilde{\xi}$ gives rise to a second stage subproblem. The stage one decision vector is included in the right-hand side of each subproblem and the resulting subproblems are tested for feasibility. If a subproblem is infeasible, a feasibility cut is generated and appended to the stage one problem, which is solved again, and the new solution passed back to the subproblems. Once a stage one solution has been found for which all subproblems are feasible, the subproblems are solved for optimality, and optimality cuts generated to force the stage one problem to account for the costs at stage two. The algorithm continues to pass stage one solutions down to the subproblems, and cuts back to the stage one problem, until it converges to the optimal solution (or proves infeasibility).

The computational effort of the L-shaped method is reduced if the problem can

be formulated to have relatively complete recourse, which removes the need to test for feasibility at stage two. Failing this, it may be possible to substitute one (or a few) problems for the full set of subproblems (Wets 1974). Birge & Louveaux (1988) extended the L-shaped algorithm to place several cuts at each iteration, rather than the single cut per iteration of the original formulation.

2.2.3 The Multi-stage Stochastic Problem

The multi-stage stochastic problem occurs when decisions must be made at each of a series of stages, and the environment faced by the decision maker in each successive stage is a function of the decisions made at previous stages and the outcomes of one or more stochastic events. At the time of making the first stage decision(s), the immediate environment is known with certainty, while future events are predicted to occur with various probabilities. For the first stage decision to be optimal it must not render any later stage infeasible, and the influence of the first stage decision on later events, must be, in some defined way, "best". The definition of what is "best" can be a major challenge, especially if the stochastic future includes realisations that are of very low probability, but of very great importance. However, the generally accepted measure is to maximise (or minimise) the expected value over all possible outcomes.

If the time horizon is finite, the DEP of the multi-stage stochastic problem can be formulated as follows:

$$
\begin{array}{lllllll}
\text{maximise} & \mathbf{cx} & + \sum_{k=1}^{K_1} p_1^k \mathbf{q}_1^k \mathbf{y}_1^k & + \sum_{k=1}^{K_2} p_2^k \mathbf{q}_2^k \mathbf{y}_2^k + \cdots & + \sum_{k=1}^{K_T} p_T^k \mathbf{q}_T^k \mathbf{y}_T^k & & \\
\text{subject to} & A\mathbf{x} & & & = \mathbf{b}_0 & & \\
& B_{10}^k \mathbf{x} + W_1^k \mathbf{y}_1^k & & & = \mathbf{b}_1^k & \text{for } k = 1 \ldots K_1 & \\
& B_{20}^k \mathbf{x} + B_{21}^k \mathbf{y}_1(a^k) & + W_2^k \mathbf{y}_2^k & & = \mathbf{b}_2^k & \text{for } k = 1 \ldots K_2 & \\
& B_{30}^k \mathbf{x} + B_{31}^k \mathbf{y}_1(a^k) & + B_{32}^k \mathbf{y}_2(a^k) & + W_3^k \mathbf{y}_3^k = \mathbf{b}_3^k & \text{for } k = 1 \ldots K_3 & & \\
& \vdots & & & \vdots & & \\
& B_{T0}^k \mathbf{x} + B_{T1}^k \mathbf{y}_1(a^k) & + B_{T2}^k \mathbf{y}_2(a^k) + \cdots & & & & \\
& \quad \cdots + B_{TT-1}^k \mathbf{y}_{T-1}(a^k) & & + W_T^k \mathbf{y}_T^k = \mathbf{b}_T^k & \text{for } k = 1 \ldots K_T & & \\
& \mathbf{x} \geq 0 \quad \mathbf{y}_t^k \geq 0 & \text{for } k = 1 \ldots K_t, & t = 1 \ldots T & & &
\end{array}
\tag{2.6}
$$

where: K_t = number of branches in the decision tree at stage t

p_t^k = probability of being on branch $k \in K_t$

W_t^k = constraint coefficients of the decision to be taken at stage t, on branch $k \in K_t$

B_{t0}^k = effects at stage t of the initial decision, $\mathbf{x}$

B_{tr}^k = effects at stage t of the decision taken at stage r

$y_r(a^k)$ = decision vector at the stage r ancestor node of branch k

This problem is extremely large, and its size is normally reduced by the introduction of state variables, so that only the effects of the decisions at the immediate ancestor node need be considered at each stage. This has the effect of reducing the number of decision variables in the subproblems, but at the cost of the addition of state variables. The introduction of state variables modifies the lower block-triangular matrix structure of problem (2.6) into the equivalent staircase form of problem (2.7):

$$
\begin{array}{lllllll}
\text{maximise} & \mathbf{cx} & + \sum_{k=1}^{K_1} p^k \mathbf{q}_1^k \mathbf{y}_1^k + \sum_{k=1}^{K_2} p^k \mathbf{q}_2^k \mathbf{y}_2^k & + \cdots & + \sum_{k=1}^{K_T} p^k \mathbf{q}_T^k \mathbf{y}_T^k & & \\
\text{subject to} & A\mathbf{x} & & & = \mathbf{b}_0 & & \\
& B_1^k \mathbf{x} & + W_1^k \mathbf{y}_1^k & & = \mathbf{b}_1^k & \text{for } k = 1 \ldots K_1 & \\
& & + B_2^k \mathbf{y}_1^k(a^k) & + W_2^k \mathbf{y}_2^k & = \mathbf{b}_2^k & \text{for } k = 1 \ldots K_2 & \\
& & & + B_3^k \mathbf{y}_2(a^k) & + W_3^k \mathbf{y}_3^k = \mathbf{b}_3^k & \text{for } k = 1 \ldots K_3 & \\
& & \ddots & & \vdots & & \\
& & & + B_T^k \mathbf{y}_{T-1}^k(a^k) & + W_T^k \mathbf{y}_T^k = \mathbf{b}_T^k & \text{for } k = 1 \ldots K_T & \\
& \mathbf{x} \geq 0, \ \mathbf{y}_t^k \geq 0 \text{ for } k = 1 \ldots K_t, & & t = 1 \ldots T & & &
\end{array}
\tag{2.7}
$$

where: B_t^k calculates the state variables from the position at the ancestor node

$y_{t-1}(a^k)$ is the decision vector at the immediate ancestor node of branch k

This problem is normally further simplified to reduce its size, and to convert it to a structure that can be exploited by customised solution techniques. The first simplification is to assume fixed recourse, which has the effect of limiting the uncertainties in the subproblem constraints to the right-hand sides. When the additional assumption is made that the objective function coefficients are deterministic, the subproblems at a given stage differ only in their right-hand sides, and this structure can be exploited by solution algorithms, (e.g. Wets 1988, Haugland & Wallace 1988, Gassmann 1990, Gassmann & Wallace 1996).

The staircase structure of problem (2.7) has been exploited to decompose the

problem into a sequence of two-stage problems. With the exception of the first and last stages, every stage is the first stage and the second stage of a pair of two-stage problems, and each of these problems can be analysed using the L-shaped algorithm. Solutions are passed down the decision tree and cuts are passed back up the tree. Birge (1988) developed an implementation of these ideas that could be applied to problems with up to three periods, and up to three hundred and seventy-five scenarios. Gassmann (1990) developed the computer code MSLiP, which also applied the L-shaped algorithm to the multi-stage problem. He assumes that the constraint matrices, W_t^k , and the objective function coefficients are deterministic, so that the subproblems differ only in their right-hand sides, and this is exploited by use of the bunching and trickling down techniques. Gassmann states that MSLiP is able to "easily handle an arbitrary number of periods" by the use of improved data structures.

Because the problem includes many subproblems that can be solved independently of each other, there is potential to use several computers in parallel to solve these subproblems simultaneously. Dantzig & Glynn (1990) and Rosen & Maier (1990) report on the use of parallel processing for solving very large stochastic problems.

2.2.4 The Value of the Stochastic Solution

Birge (1982),(see also Birge 1995, Birge 1997, Birge & Louveaux 1997) introduced the idea of the *value of the stochastic solution* (VSS). The VSS is intended to provide a measure of the value to the decision maker of solving a stochastic optimisation formulation rather than the computationally more tractable expected value formulation in which the uncertain parameters are represented by their expected values. In the expected value formulation the problem is reduced to a deterministic problem by summarising the uncertainties into a single scenario.

In Birge (1982) and Birge (1995), the stochastic program with recourse is referred to as the *recourse problem* and its solution as the **RP** solution. In Birge (1997) it is referred to as the *here and now* solution **HN**, because the decisions must be taken 'here and now', before the uncertainty is resolved. If the decision maker had access to perfect information, then the decision at each stage could be taken after the uncertainty has been resolved, and this is called the *wait and see* solution **WS**. In

Birge (1982) the solution to the mean value problem is called the **EV** solution, while in Birge (1997) it has become the *mean value* solution **MV**. Finally, the *expectation of the mean value solution* **EMV** (or **EEV**) is defined to be the objective function value obtained when the mean value solution is used as the solution to the stochastic programming formulation (the *recourse* or *here and now* problem). That is, the EMV is the expected objective function value of problem (2.1) when $\mathbf{x}$ is fixed to be the mean value solution.

The VSS is defined to be the difference between the here and now solution and the expectation of the mean value solution:

$$VSS = HN - EMV$$

The expected value of perfect information (EVPI) is defined to be the difference between the wait and see solution and the here and now solution:

$$EVPI = WS - HN$$

Birge suggests that the VSS can be used in a similar fashion to the expected value of perfect information (EVPI). The EVPI gives an upper bound on the expected value of obtaining additional information and so reducing the uncertainty. Thus it provides an upper bound on the amount that should be spent to obtain additional information. In Birge (1982) it is proposed that the VSS be calculated to provide an upper bound on the amount that should be spent to solve the stochastic optimisation problem rather than the expected value problem. A method is developed for finding bounds for the VSS by solving a series of subproblems that are more tractable than the full stochastic optimisation problem.

The VSS has come to be used as a measure of the value of solving the stochastic optimisation formulation of a problem, rather than its expected value formulation. However, the VSS may overstate the value of solving the stochastic optimisation formulation, because the EMV solution is evaluated under the scenarios of the here and now formulation, and these are the scenarios that were used to find the stochastic solution. Given that these scenarios are themselves only representations of the uncertainty, it would seem appropriate to use the more rigourous measure of the value of solving the stochastic problem that would be obtained by testing both the EMV solution, and the stochastic solution under a new set of scenarios. It is pertinent to note that the expected value scenario is a possible scenario (at least

in most problems), and when the two solutions are compared under this scenario, the EMV solution will do better than stochastic one. The point is that the VSS is calculated by comparing the two solutions under the scenarios that were used to find one of them, and so the result is biased towards that solution.

2.3 Other Approaches to Handling Uncertainty

2.3.1 Single Scenario Approaches

The decision maker may attempt to resolve the uncertainty before the fact by obtaining a forecast. The situation can then be formulated as a deterministic problem and solved to find the optimal decision. Clearly this approach is only tenable if the decision maker has confidence in the forecast, although sensitivity analysis can be used to evaluate the robustness of the solution.

Another commonly used method of reducing the uncertainty to a single scenario is to solve the *expected-value problem.* If the uncertain parameters appear in the objective function only, then the solution to the expected-value problem is the optimal stochastic solution. However, if uncertain parameters appear in the constraints, then the expected-value formulation does not produce the optimal stochastic solution. Although the expected-value decision may be close to the optimal stochastic solution for some problems, the reported objective function value will normally be overstated. These shortcomings have been widely discussed in the stochastic optimisation literature, (e.g. Wets 1983, Birge 1995, Birge 1997). As discussed in Section 2.2.4, the value of the stochastic solution (VSS) has been developed as a measure of how much could be gained by solving the stochastic problem, rather than the expected-value problem.

2.3.2 Scenario Analysis

When a set of quantitative scenarios is available, deterministic optimisation can be used to find the optimal solution for each scenario individually. If the same decision is optimal for all scenarios, then the problem has been solved, although this probably means that the scenarios are an inadequate representation of the future uncertainty. Normally, each scenario will have a different optimal solution, and each scenario-optimal solution will perform poorly under at least one of the other scenarios. To

reduce the downside risk, the decision maker may want to find a compromise decision that performs adequately under all scenarios. This can be attempted by a trial and error process, in which the decision maker constructs candidate decisions and tests them under each of the scenarios. This process continues until a satisfactory compromise is found, or the decision maker decides to stop. The disadvantages of this approach are the lack of a systematic method for developing new decisions, and the danger that the decisions tested are dominated by other decisions that were not considered.

The shortcomings of scenario analysis are discussed in Rockafellar & Wets (1991) and Wets (1989), in which scenario aggregation is proposed. Scenario aggregation is reviewed in Section 2.7.3.

2.4 Probabilistic Programming

2.4.1 Chance-Constrained Programming

Charnes & Cooper (1963) proposed the chance-constrained approach to stochastic programming using a different model structure to that of the stochastic program with recourse. In the chance-constrained progamming model, all decisions must be implemented before the uncertainty is resolved, and there is no recourse available for making adjustments after the outcomes are observed. However, rather than requiring that all constraints be met under every outcome, chance-constrained progamming requires that the stochastic constraints be met with specified probability levels.

Charnes and Cooper start with the general form of an LP:

maximise	$\mathbf{cx}$
subject to	$A\mathbf{x} \leq \mathbf{b}$

where: some, or all, elements of $\mathbf{c}$, A and $\mathbf{b}$ may be random variables

and, assuming that matrix A is deterministic, reformulate it as the chance-constrained

problem:

$$\begin{aligned} &\text{maximise} && \mathrm{E}(\mathbf{c})\mathbf{x} \\ &\text{subject to} && \mathrm{P}(a_i\mathbf{x} \leq b_i) \geq \alpha_i \quad \forall\, i \\ & && \mathbf{x} = D\mathbf{b} \end{aligned} \tag{2.8}$$

where: $\mathrm{E}(\mathbf{c})$ = the expectation of the stochastic objective vector $\mathbf{c}$

a_i = i^{th} row of the constraint matrix A

D = an $n \times m$ matrix, referred to as the "decision rule"

α_i = the prescribed probability that the right-hand side, b_i, be met

α_i = 1 for deterministic contraints

Charnes and Cooper then show that there is a deterministic equivalent:

$$\begin{aligned} &\text{maximise} && \mathrm{E}(\mathbf{c})D\mathrm{E}(\mathbf{b}) \\ &\text{subject to} && \mu_i(D) - v_i \geq 0 \quad \forall\, i \\ & && -K^2_{\alpha_i}\sigma^2_i(D) + K^2_{\alpha_i}\mu^2_i(D) + v^2_i \geq 0 \quad \forall\, i \\ & && v_i \geq 0 \quad \forall\, i \end{aligned} \tag{2.9}$$

where: $\sigma^2_i(D) = \mathrm{E}(a_iD\mathbf{b} - b_i)^2$

$\mu^2_i(D) = \mathrm{E}(\mu_{bi} - a_iD\mu_b)^2$

K_{α_i} = the cumulative distribution of $(a_iD\mathbf{b} - b_i)$

This formulation is generally reported as:

$$\begin{aligned} &\text{maximise} && \mathrm{E}(\mathbf{c})\mathbf{x} \\ &\text{subject to} && A_i\mathbf{x}_i \leq \mathbf{k}_i \quad \forall\, i \end{aligned} \tag{2.10}$$

where $\mathbf{k}_i$ corresponds to the α_i percentage of the cumulative density function of $\mathbf{b}_i$.

When deriving this deterministic equivalent in their (1963) paper, Charnes and Cooper make the following assumptions:

1. The choice of values for $\mathbf{x}$ does not disturb the densities associated with the random variables $\mathbf{b}$ and $\mathbf{c}$.

2. The vectors $\mathbf{b}$ and $\mathbf{c}$ are not correlated, although their respective components may be.

3. The frequency distributions for the variates $(a_i D\mathbf{b} - b_i)$ are symmetric and the distributions associated with these variates are completely specified by their first two moments. They remark that this does not mean that the distributions for $\mathbf{b}$ and $\mathbf{c}$ must have these properties.

4. To provide a specific instance to discuss they assume that $(a_i D\mathbf{b} - b_i)$ is normally distributed.

5. Finally they assume $\alpha_i > 0.5$ so that $K_{\alpha_i} \geq 0 \quad \forall\, i$

Given these assumptions, Charnes and Cooper demonstrate that (2.9) is a convex programming problem in variables D and $\mathbf{v}$.

Although the constraints include squared terms, these are derived by squaring inequalities in which all terms are known to be nonnegative. This means that, for many problems, the constraints can be formed by taking the positive square roots of these terms, and so making the problem linear. This is the approach taken in the example at the end of their paper.

Charnes, Cooper & Symonds (1958) report an application of chance-constrained programming to the supply of heating oil to a statistically described demand (via weather dependency). Guldmann (1983) uses chance-constrained programming to explore trade-offs available to natural gas distribution utilities who must meet a stochastic demand for gas.

Miller & Wagner (1965) extended the approach to constrain the joint probability of meeting all random right-hand sides to be at least some specified constant. They assume that the $\mathbf{b}_i$ are stochastically independent and formulate the problem as:

$$\begin{aligned} &\text{maximise} && \mathrm{E}(\mathbf{c})\mathbf{x} \\ &\text{subject to} && \prod_i \mathrm{P}_i\Big(A_i\mathbf{x} \leq b_i\Big) \geq \theta \end{aligned} \tag{2.11}$$

where θ is a scalar. This is a nonlinear program, with no linear, deterministic equivalent.

Because this formulation is nonlinear, the problem of local optima is relevant. Miller and Wagner develop a nonlinear transformation of the joint chance constraint that yields a concave function (and so a convex programme) for many distributions, including the normal, and certain cases of the gamma and Weibull. They take the

natural logarithm of both sides to yield:

$$\sum_i \ln(P_i) \geq \ln \theta$$

Another variation on the chance-constrained approach is to constrain the expected value of the number of right-hand sides that are met to be at least some specified value (see Sengupta 1972). This is equivalent to constraining the total of the probabilities of meeting the right-hand sides to be at least some specified constant:

$$\begin{aligned} &\text{maximise} && \mathrm{E}(\mathbf{c})\mathbf{x} \\ &\text{subject to} && \sum_i \mathrm{P}_i\Big(A_i\mathbf{x} \leq b_i\Big) \geq \phi \end{aligned} \tag{2.12}$$

where ϕ is a scalar. Unless the $\mathbf{b}_i$ are uniformly distributed, this is a nonlinear programme, with no linear, deterministic equivalent.

2.4.2 Chance-Maximising Programming

Hof & Pickens (1991) propose a rather different approach in which the probability of meeting the random right-hand sides is maximised, rather than constraining the problem to meet minimum required probabilities. They suggest that this formulation is appropriate for renewable resource management problems, in which "feasibility (meeting output targets and/or not exhausting input availability) may often be a very high priority". They propose three approaches, each being a counterpart to one of the chance-constrained approaches discussed above. The first is to maximise the minimum probability of meeting a right-hand side, the maxi-min formulation:

$$\begin{aligned} &\text{maximise} && \lambda \\ &\text{subject to} && \lambda \leq \mathrm{P}_i\Big(\sum_j a_{ij}x_j \leq b_i\Big) \qquad \forall\, i \end{aligned} \tag{2.13}$$

where λ is a scalar. The objective function is ignored. However, a target value could be specified for the objective, and achievement of this target included as one of the constraints.

They also formulate a joint probability maximising formulation, and one that maximises the sum of the probabilities of meeting the right-hand sides. These formulations are applied to an example from the forestry industry, in which a forest

harvest programme must be determined for three types of forest land so that stochastic output targets are met for three different products. The model includes hard constraints on land use and road construction, while the achievement of the output targets is expressed as probabilistic constraints. The objective is to maximise the return from the harvest programme while meeting the output targets with a specified probability.

Hof & Pickens (1991) compared the solutions produced by the different chance-constrained and chance-maximising formulations. They concluded that the Charnes and Cooper approach is suitable when a specified level of confidence of attaining individual right-hand sides is desired, but that it can lead to inequitable levels of risk across the right-hand sides. If a more equitable attainment is desired, then the chance-maximising approaches are more appropriate. Because these approaches can produce substantially different results the authors suggest that several of them should be used to analyse a problem.

2.4.3 Extensions to Include a Stochastic A Matrix

In Hof et al. (1992) the chance-constrained and chance maximisation problems are extended to include a stochastic A matrix. The formulation accounts for correlated terms within rows of the A matrix, which means that terms within a particular constraint can share sources of randomness. However, they assume that the rows are independent, stating that "Relaxation of this assumption is, at this time, analytically intractable." They discuss the validity of this assumption, and conclude that it is problem dependent. They also conclude that the impact of ignoring between-row covariances is difficult to predict. The chance-constrained and chance-maximising formulations are applied to the same model as in Hof & Pickens (1991), with modifications to include stochastic terms in the A matrix. Again they conclude that the different approaches can yield substantially different results. They also comment that "The nonlinear programming approaches presented are clearly only applicable to small problems, and local optima are an ever-present concern. If nonlinear programming solution capabilities continue to develop, however, the potential application of these approaches will develop as well.".

It seems that chance-constrained and chance-maximising programming are appropriate for situations in which the uncertainty is not resolved in time for recourse

to be possible. The forest harvesting example used in Hof & Pickens (1991) has been formulated in this manner, presumably because information about yields obtained from the stands that are harvested do not resolve the uncertainty about the yields obtainable from the stands not yet cut. This means that there is no resolution of the uncertainty, and thus no opportunity to take recourse action part way through the harvest.

For situations in which the uncertainty will be resolved in time for recourse actions to be implemented, and in which the decision process is sequential, it seems that stochastic programming with recourse is more appropriate than chance-constrained programming.

The data requirements of chance-constrained programming are also demanding, in that the uncertain parameters must be described in terms of cumulative probability distributions. For the forest harvesting examples used by Hof et al. (1992) reliable data for the yields from the different types of forest land are likely to be available from historical data. However, in strategic planning problems historical data is either unavailable, or it is not a reliable predictor of the future. This means that probabilistic programming is not suitable for analysing strategic planning problems.

2.5 Stochastic Dynamic Programming

Dynamic programming (due to Bellman 1957) is designed to find values for a sequence decisions, in order to optimise an objective function. The system must be described by a small number of state variables, such that the decision at any stage can be chosen without knowledge of the decisions that were taken to reach that stage. In conventional dynamic programming the state variable must be discrete, and any decision at any stage must lead to one of the discrete values of the state variable at the next stage. For problems in which the state variable is naturally discrete, dynamic programming will produce an optimal policy. However, if the state variable is continuous it must be approximated by a grid of discrete values, and dynamic programming can no longer be relied upon to produce an optimal solution. An attraction of deterministic dynamic programming is that, in the course of finding the optimal policy, the optimal decision for every value of the state variable at every

stage in the planning horizon is identified. This is in contrast to the solution produced by a deterministic linear programming formulation, which specifies a single state-decision pair for each stage. This means that dynamic programming solutions can provide different insights into the problem than are available from LP solutions, and that the problem does not have to solved for a new solution as soon as events deviate from the predicted outcomes.

In deterministic dynamic programming it is assumed that every decision, at every stage, will lead to one, and only one, value of the state variable at the next stage. In stochastic dynamic programming a decision may lead to any one of several values of the state variable at the next stage, and the value actually observed cannot be predicted. The stochastic problem is formulated using a discrete probability distribution to describe the probability that a decision will lead to a particular value of the state variable at the next stage. The optimal solution of a stochastic dynamic programme consists of a sequence of conditional decisions at each stage, one for each value of the state variable at that stage that may be reached after implementing the optimal decision at the first stage.

In deterministic dynamic programming the value that the state variable will assume at each stage, if the optimal policy is implemented, can be determined in advance for all stages. In contrast, the solution to the stochastic dynamic programming problem can only show the values that may occur at each stage if the optimal policy is implemented.

The difficulties involved in making continuous state variables discrete have been addressed by the development of dual dynamic programming (Read & George 1990). In this formulation a single product is produced and stored from period to period. The production options in each period are described by a linear subproblem which determines the optimal process mix for a given level of output. The subproblem is solved parametrically for each period to find the marginal costs of production over the feasible range of output. These marginal production costs form a "supply curve" for the product in each period. This "supply curve" is a step function, in which each step corresponds to a marginal cost, and the width of the step is the range of output levels to which this marginal cost applies. An end-of-horizon inventory value function is required as a starting point for the backwards recursion. The backwards recursion uses the marginal production costs in the last period to project the end-of-horizon inventory value function back to the beginning of the last

period. This forms the marginal value curve for the end of the next period, and the recursion is repeated until the beginning of the horizon is reached.

Dual dynamic programming overcomes the shortcomings involved in representing a continuous state variable as a discrete state variable, because the values chosen to form the grid are the critical values at which the marginal cost of production changes, rather than being arbitrary values. This means that all stock levels between two points on the grid have the same marginal cost. If the grid was chosen arbitrarily, two adjacent stock levels may have different marginal costs, and it is impossible to know which cost applies to a stock level between them, or even if the intermediate point has a marginal cost that is different from the costs of them both.

Read & Boshier (1989) extend dual dynamic programming to the stochastic case.

Pereira (1989) proposes a decomposition approach to solving multi-stage optimisation problems that he calls "dual dynamic programming", but it is not the same as the dual dynamic programming of Read and George. In fact, Pereira's approach is an application of the L-shaped algorithm to the multi-stage problem, in which locally accurate piecewise approximations of the marginal value functions are found by the generation of optimality cuts. Pereira's "dual dynamic programming" formulation finds a single optimal trajectory of reservoir levels, and the within-period decisions required to move between them. This means that the problem has to be solved again when the observed reservoir levels deviate from the optimal strategy. On the other hand, Read and George's dual dynamic programming formulation finds the optimal within-period decisions for all reservoir levels, in all stages, which means that their solution provides a great deal more information about the problem situation than Pereira's does. It is, however, much more difficult to apply to problems with a state space of more than one or two dimensions.

Stochastic dynamic programming has been widely applied to strategic planning decision making, particularly in the energy sector. Dapkus & Bowe (1984) build a stochastic dynamic programming model to investigate capacity expansion options for a hypothetical utility that could add a combination of three technologies to its existing capacity. They use an eighteen year planning horizon, broken up into three, six-year stages, and solve four models to provide comparisons. The first model is a deterministic problem, and the remaining three are stochastic. The second model includes uncertainty in demand with three possible outcomes at the beginning of stage 2, and the beginning of stage 3, producing nine scenarios. The third model

includes uncertainty in the commercial availability of an alternative technology (fuel cells), with two outcomes at stage 2 and stage 3, producing four scenarios. Finally, uncertainties in demand and technologies are included, but it is not clear from the paper how many scenarios are produced. The paper concludes that the inclusion of uncertainty in the model changes the recommended expansion plan away from the deterministic solution. In this example, additional capacity is built to hedge against high demand and low availability, and technologies with unpredictable futures are avoided.

Haugen (1996) applies stochastic dynamic programming to the problem of scheduling development of Norway's natural gas fields to supply the European market. The market is viewed as a contractual market, in which a supplier contracts to supply specified quantities of gas each year over a given planning horizon. Failure to meet the contract quantities exactly, both over supply and under supply, are penalised in the pricing structure of the contracts. This paper concentrates on the resource uncertainty, that is, the length of time each field will be able to produce at full capacity, before output starts to tail off. This uncertainty is due to imprecision in the estimates of the quantities of gas in the fields, and the unpredictability of extraction technologies which affect the portion of the gas that will actually be recoverable. The problem is formulated with seven time periods, and five gas fields, one of which is known with certainty. The principal insight that seems to come out of this paper is that, as the number of alternatives is increased (in this case the number of stages in which development of a field can start), the "curse of dimensionality" has a drastic effect on the solution times. Some possible remedies are suggested as avenues for future research.

2.6 Fuzzy, Grey and Interval Programming

In the previous sections we have discussed modelling techniques in which it is assumed that the behaviour of the uncertain parameters can be described using probability distributions. That is, an uncertain parameter will take one of several discrete values, or take a value from a continuous interval, according to some probability distribution. In this section we will briefly consider approaches in which the uncertain parameters have a single value, but that value is not known precisely. These parameters are variously described as being *fuzzy, grey* or *interval.* Various methods

of analysing problems with imprecise data are associated with each of these terms. There are many similarities between these methods, but it appears that three literatures have evolved independently with little reference to one another. For this reason the methods associated with each of the three terms will be discussed separately.

2.6.1 Fuzzy Decision Making

Fuzzy set theory and its applications has given rise to a very large literature, which will be discussed only briefly in order to relate it to stochastic programming. A parameter is said to be fuzzy if it can be considered to have a value, but it is unclear exactly what that value is. A fuzzy number is not a random variable, it does not take one of several values according to some probability distribution, instead it has some value, but that value is not precisely known. Bellman & Zadeh (1970) provide the following descriptions of the differences between fuzziness and randomness.

> Our contention is that there is a need for differentiation between *randomness* and *fuzziness*, with the latter being a major source of imprecision in many decision processes. By fuzziness we mean a type of imprecision which is associated with *fuzzy sets*, that is, classes in which there is no sharp transition from membership to nonmembership.
>
> What is the distinction between randomness and fuzziness? Essentially, randomness has to do with uncertainty concerning membership or nonmembership of an object in a nonfuzzy set. Fuzziness, on the other hand, has to do with classes in which there may be grades of membership intermediate between full membership and nonmembership.

Bellman & Zadeh give the following definition of a fuzzy set:

> *Definition* Let $X = \{x\}$ denote a collection of objects (points) denoted generically by x. Then a *fuzzy set* A *in* X is a set of ordered pairs
>
> $$A = \{(x, \mu_A(x))\}, \qquad x \in X$$
>
> where $\mu_A(x)$ is termed the *grade of membership of* x *in* A, and $\mu_A : X \to M$ is a function from X to a space M called the *membership space*. When M contains only two points, 0 and 1, A is nonfuzzy and its membership becomes identical with the characteristic function of a nonfuzzy set.

In what follows, we shall assume that M is the interval $[0,1]$, with 0 and 1 representing, respectively, the lowest and highest grades of membership. Thus, our basic assumption is that a fuzzy set A—despite the unsharpness of its boundaries—can be defined *precisely* by associating with each object x a number between 0 and 1 which represents its grade of membership in A. A fuzzy set is said to be normal (or normalized) if the grade of membership of at least one member of the set is 1.

Example Let $X = \{0, 1, 2, \dots\}$ be a collection of nonnegative integers. In this space, the fuzzy set A of "*several* objects" may be defined (subjectively) as the collection of ordered pairs

$$A = \{(3, 0.6), (4, 0.8), (5, 1.0), (6, 1.0), (7, 0.8), (8, 0.6)\}$$

with the understanding that we list only those pairs $(x, \mu_A(x))$ in which $\mu_A(x)$ is positive.

Bellman & Zadeh then discuss additional concepts of fuzzy set theory, and proceed to discuss fuzzy goals, constraints and decisions. Goals are said to be representations of statements such as "decision variable x should be substantially larger than 10", while constraints are said to be representations of statements such as "x should be approximately between 2 and 10". A possible membership function for the goal is given as

$$\begin{aligned} \mu_G(x) &= 0, & x < 10, \\ &= (1 + (x - 10)^{-2})^{-1}, & x \geq 10 \end{aligned}$$

and the constraint as

$$\mu_C(x) = (1 + a(x - 6)^m)^{-1}$$

where a is a positive number, and m is a positive even integer.

When these definitions of goals and constraints are used, they can be treated identically, and the membership function of the fuzzy set of alternative decisions, D, is defined by the intersection of the fuzzy sets that describe the goals and constraints.

The membership function of the intersection of two fuzzy sets, A and B, is defined as:

$$\mu_{A \cap B}(x) = \text{Min}(\mu_A(x), \mu_B(x)), \qquad x \in X$$

or, more concisely

$$\mu_{A \cap B} = \text{Min}(\mu_A, \mu_B)$$

Which means that the membership function of the fuzzy set of decisions is given by

$$\mu_D = \text{Min}(\mu_{G_1}, \mu_{G_2}, \ldots, \mu_{G_n}, \mu_{C_1}, \mu_{C_2}, \ldots, \mu_{C_m})$$

Thus, the solution to a fuzzy problem is, itself, fuzzy. However, the decision maker must select a decision to implement, and Bellman & Zadeh suggest that "it is reasonable in many instances to choose that x or x's which have maximal grade of membership in D."

They then go on to discuss multistage decision processes in fuzzy environments, which they solve using dynamic programming. They extend these ideas to stochastic systems in a fuzzy environment, in which they maximise the expectation of the membership function of the fuzzy set of alternative decisions. This problem is also solved using dynamic programming.

2.6.2 Fuzzy Linear Programming

In fuzzy linear programming, some, or all, of the objective function coefficients, the constraint matrix coefficients, and the right hand sides are modelled as fuzzy sets.

A general model of a fuzzy linear programming problem (FLP-problem) is represented by:

$$\begin{aligned} & \tilde{C}_1 x_1 \oplus \tilde{C}_2 x_2 \oplus \ldots \oplus \tilde{C}_n x_n \quad \rightarrow \tilde{\text{Max}} \\ \text{subject to} \quad & \tilde{A}_{i1} x_1 \oplus \tilde{A}_{i2} x_2 \oplus \ldots \oplus \tilde{A}_{in} x_n \tilde{\leq} \tilde{B}_i, \qquad i = 1, \ldots, m \\ & x_1, x_2, \ldots, x_n \geq 0 \end{aligned} \tag{2.14}$$

Where $\tilde{A}_{ij}$, $\tilde{B}_i$, $\tilde{C}_j$, $i = 1, \ldots, m$; $j = 1, \ldots, n$, are fuzzy sets in $\mathcal{R}$. Some of the coefficients may be known exactly, in which case some of the fuzzy sets may actually be nonfuzzy numbers. The symbol $\oplus$ represents *extended addition* by which the left hand side of a fuzzy constraint can be aggregated to a fuzzy set.

Rommelfanger (1996) presents a survey of methods for solving fuzzy linear programs, which require that the uncertain parameters be modelled as fuzzy numbers or fuzzy intervals of the L-R-type. (A description of the L-R-type follows.)

A *fuzzy number* is defined as being a convex, normalised fuzzy set
$\tilde{A} = \{(x, \mu_A(x)) | x \in \mathcal{R}\}$ on the real line $\mathcal{R}$ if

i) there exists exactly one $x_0 \in \mathcal{R}$ with the membership degree $\mu_A(x) = 1$, and

ii) $\mu_A(x)$ is piecewise continuous in $\mathcal{R}$

If, for $\tilde{A}$, there exists more than one $x \in \mathcal{R}$ with a membership degree $\mu_A(x) = 1$, then the set is called a *fuzzy interval.*

A fuzzy number $\tilde{N} = \{(x, \mu_N(x)) | x \in \mathcal{R}\}$ is said to be of the L-R-type if there exist reference functions L and R and scalars $\alpha, \beta > 0$ such that

$$\mu_N(x) = \begin{cases} L((n-x)/\alpha) & \text{if} \quad x < n \\ R((x-n)/\beta) & \text{if} \quad x \geq n \end{cases}$$

Symbolically $\tilde{N}$ is denoted by $(n, \alpha, \beta)_{LR}$

A fuzzy interval $\tilde{M} = \{(x, \mu_M(x)) | x \in \mathcal{R}\}$ is said to be of the L-R-type if there exist reference functions L and R and scalars $\alpha, \beta > 0$ such that

$$\mu_M(x) = \begin{cases} L((m_1-x)/\alpha) & \text{if} \quad x < m_1 \\ 1 & \text{if} \quad m_1 \leq x \leq m_2 \\ R((x-m_2)/\beta) & \text{if} \quad x \geq m_2 \end{cases}$$

Symbolically $\tilde{M}$ is denoted by $(m_1, m_2, \alpha, \beta)_{LR}$

Rommelfanger discusses the issue of modelling fuzzy data, and how to derive appropriate membership functions from the data that a decision maker is able to provide. For fuzzy right-hand sides he suggests that the decision maker provide a value, b_i, which has a membership value of 1. For example, this could be the upper limit for the amount of that resource that the decision maker can guarantee will be available. This appears to correspond to the idea of the 'worst case' scenario. The decision maker then decides on a second (larger) value, b_i^ϵ, and it is assumed that all values outside the range defined by these two values have membership values of zero. The decision maker may refine the membership function by defining an intermediate value, b_i^λ, so that the membership line becomes piecewise linear. See Figure 2.1. This membership function is symbolised by

$$\tilde{B}_i = (b_i; 0, 0; \beta_i^\lambda, \beta_i^\epsilon)^{\lambda,\epsilon} \qquad \text{where} \quad \beta_i^\lambda = b_i^\lambda - b_i \quad \text{and} \quad \beta_i^\epsilon = b_i^\epsilon - b_i$$

A similar approach is used to determine fuzzy intervals for the objective function coefficients, and for the left-hand side coefficients of the constraints. The decision maker specifies a range of values that have the highest chance of realisation, and

the values in this range are assigned a membership value of 1. On each side of this range the decision maker specifies a value that is considered to have a sufficiently low chance of realisation for it to be ignored. These reductions of the membership value from 1 to 0 may be expressed as piecewise linear functions by specifying intermediate values, as was done for the right-hand sides. See Figure 2.2. However, Rommelfanger reports that the use of the intermediate values $\underline{a}_{ij}^{\lambda}$ and $\overline{a}_{ij}^{\lambda}$ is often dispensed with. This membership function is symbolised by

$$\tilde{\mathcal{A}}_{ij} = (\underline{a}_{ij}; \overline{a}_{ij}; \underline{\alpha}_{ij}^{\epsilon}; \overline{\alpha}_{ij}^{\epsilon})^{\epsilon} \qquad \text{where} \quad \underline{\alpha}_{ij}^{\epsilon} = \underline{a}_{ij} - \underline{a}_{ij}^{\epsilon} \quad \text{and} \quad \overline{\alpha}_{ij}^{\epsilon} = \overline{a}_{ij}^{\epsilon} - \overline{a}_{ij}$$

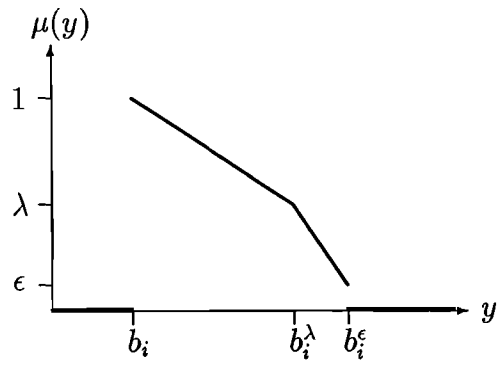

Figure 2.1:
Membership Function of a RHS
Point b_i is the largest value for the right-hand side of constraint i that the decision maker is certain will be available. b_i^{λ} is a value at which the slope of the membership function changes, and b_i^{ϵ} is the maximum value for the right-hand side that the decision maker thinks can reasonably be expected.

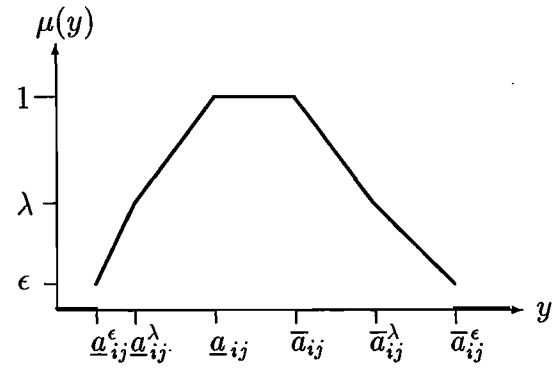

Figure 2.2:
Interval Membership Function
The values in range $\underline{a}_{ij}$ to $\overline{a}_{ij}$ are those considered to be the most likely to occur. The values $\underline{a}_{ij}^{\epsilon}$ and $\overline{a}_{ij}^{\epsilon}$ are the minimum and maximum values that the decision maker thinks should be considered. $\underline{a}_{ij}^{\lambda}$ and $\overline{a}_{ij}^{\lambda}$ are values at which the slopes of the membership function change.

2.6.2.1 Linear Programming with Fuzzy Right-Hand Sides

The simplest form of FLP-models occur when the decision maker is able to supply nonfuzzy numbers for all coefficients, except for some of the right-hand sides. Rommelfanger refers to nonfuzzy numbers as *crisp* numbers, and we will also use that term in this discussion.

Linear programming problems with fuzzy right-hand sides of the type

$$g_i(\mathbf{x}) = g_i(x_1, x_2, \ldots, x_n) = a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \tilde{\leq} \tilde{B}_i$$

were first discussed by Zimmerman (1975) who described the imprecise right-hand side, $\tilde{B}_i$, by a fuzzy set with support $[b_i, b_i + d_i] \subseteq \mathcal{R}$, $d_i \geq 0$, and a monotone decreasing membership function μ_{B_i}.

The membership function μ_{B_i} is specified so that the function

$$\mu_i(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad g_i < b_i \\ \mu_{B_i}(g_i) & \text{if} \quad b_i \leq g_i \leq b_i + d_i \\ 0 & \text{if} \quad g_i > b_i + d_i \end{cases} \tag{2.15}$$

expresses the "individual satisfaction" of the decision maker in relation to $g_i(\mathbf{x}) = g_i(x_1, x_2, \ldots, x_n)$.

Rommelfanger reports that, according to various authors (such as Zimmerman), the inequality relation '$\tilde{\leq}$' in these constraints may be interpreted as

$$g_i(\mathbf{x}) \tilde{\leq} \tilde{B}_i \;\Leftrightarrow \begin{cases} g_i(\mathbf{x}) \leq b_i + d_i, \\ \mu_i(\mathbf{x}) \rightarrow \text{ Max}, \end{cases}$$

i.e. each soft constraint adds an additional objective to the decision problem.

This expression also says that the soft constraint has a hard upper limit that may not exceeded. In other words, it is possible to use more than b_i of resource i, although at a reduction in the decision maker's satisfaction, but additional usage is limited to be no more than d_i.

Various models for representing $\mu_i(g_i)$ on the interval $[b_i, b_i + d_i]$ have been proposed, the simplest being a linear relationship. Other authors have suggested the use of concave and s-shaped relationships, either by using a function, such as an exponential or hyperbolic function, or by making it piecewise linear.

Thus a linear programming system of the type

$$\begin{aligned} & z(\mathbf{x}) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \quad \rightarrow \text{Max} \\ \text{subject to} \quad & a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \;\tilde{\leq}\; \tilde{B}_i, \qquad i = 1, \ldots, m_1 \\ & a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \;\leq\; B_i, \qquad i = m_1 + 1, \ldots, m \\ & x_1, x_2, \ldots, x_n \geq 0 \end{aligned} \tag{2.16}$$

with m_1 soft constraints and $m - m_1$ crisp constraints may be described by a multiobjective optimisation system of the type

$$\underset{x \in X_U}{\text{Max}} \big(z(\mathbf{x}), \mu_1(\mathbf{x}), \ldots, \mu_{m_1}(\mathbf{x}) \big)$$

where

$$X_U = \Big\{\mathbf{x} \in \mathcal{R}^n | a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i + d_i, \qquad \forall\, i = 1, 2, \ldots, m_1$$
$$\text{and } a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i, \qquad \forall\, i = m_1 + 1, \ldots, m\Big\}$$

The objective function $z(\mathbf{x})$ is usually replaced by its membership function, $\mu_z(\mathbf{x})$, which can be compared with the fuzzy objective values $\mu_i(\mathbf{x})$. The basic shape of $\mu_z(\mathbf{x})$ is given by

$$\mu_z(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad z \leq \underline{z} \\ \mu_z(z) & \text{if} \quad \underline{z} \leq z \leq \overline{z}, \\ 1 & \text{if} \quad z > \overline{z} \end{cases}$$

where $\mu_z(z)$ is a monotone increasing function of z, and

$$\overline{z} = \underset{x \in X_U}{\text{Max}} \Big(z(\mathbf{x})\Big) \qquad \text{and} \qquad \underline{z} = \underset{x \in X_L}{\text{Max}} \Big(z(\mathbf{x})\Big)$$

$$\text{where } X_L = \Big\{\mathbf{x} \in \mathcal{R}^n | a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i, \quad \forall\, i = 1, 2, \ldots, m\Big\}$$

The multiobjective function becomes:

$$\underset{x \in X_U}{\text{Max}} \Big(\mu_z(\mathbf{x}), \mu_1(\mathbf{x}), \ldots, \mu_{m_1}(\mathbf{x})\Big)$$

Rommelfanger describes the membership functions as being used to measure the "satisfaction" of the decision maker. It would seem that they are, in fact, being used as utility functions, and the term utility will be used in this discussion.

The feasible set X_U corresponds to the 'best case' scenario, while X_L is the feasible set corresponding to the 'worst case' scenario. Thus the objective function of the original problem (2.16) has been redefined so that its contribution to the decision maker's utility is zero if its value falls below the 'worst case' result, and its contribution increases as its value increases, until it reaches a maximum contribution of 1 at the objective function value of the 'best case' scenario.

The objective of maximising the utility received from the value of the original objective function, $z(\mathbf{x})$, is included as one of a set of multiple objectives, where the other objectives are the utilities obtained by satisfying the constraints. The utility obtained from a constraint is at its maximum of 1 when the constraint is satisfied with its right-hand side at its minimum setting, that is, its value under the 'worst case' scenario. The utility derived from a constraint decreases as the constraint is

met at increasing values of its right-hand side. The utility to the decision maker is zero when the right-hand side exceeds the value available under the 'best case' scenario.

Thus the problem has been reformulated into a multiobjective problem in which one objective is to maximise the objective value of the original problem. The other objective is to limit the use of resources to the 'worst case' availabilities. Each constraint with a fuzzy right-hand side contributes an objective function to the problem. The objectives have all been expressed as utilities, normalised to the range $[0, 1]$ by the use of the membership functions of fuzzy sets. This formulation can be seen as using a form of penalty function, in that the terms $\mu_i(\mathbf{x})$ (the utilities of resource use) decrease with increasing values of $\mathbf{x}$. The definition of $\mu_i(\mathbf{x})$ means that once the use of resource i exceeds the value $b_i + d_i$, any increase in use would attract a penalty of zero. Use of this 'free' resource is prevented by including a hard constraint for each resource, i, with a right-hand side value of $b_i + d_i$.

Rommelfanger reports that it is generally assumed that the decision maker wants a 'compromise solution' to this problem, rather than the complete set of noninferior solutions, and that a suitable description of the decision maker's preferences is given by

$$\lambda(\mathbf{x}) = \min\left(\mu_z(\mathbf{x}), \mu_1(\mathbf{x}), \ldots, \mu_{m_1}(\mathbf{x})\right)$$

so that the problem can be formulated as the Max-Min problem

$$\begin{array}{lll} \text{maximise} & \lambda & \\ \text{subject to} & \lambda \leq \mu_z(\mathbf{x}), & \\ & \lambda \leq \mu_i(\mathbf{x}), \qquad i = 1, \ldots, m_1 & (2.17) \\ & \mathbf{x} \in X_U, \qquad \lambda \leq 1 & \end{array}$$

Provided that linear, or piecewise linear, functions are used for the membership functions, this problem can be solved using standard software. However, use of the min-operator as the preference function means that the solution to this problem may not be Pareto-optimal, and some authors suggest using an interactive approach to ensure that a Pareto-optimal solution is found. An alternative approach would be to fix the value for λ at its optimal value, and then maximise for $\mu_z(\mathbf{x})$. If a better value is found for $\mu_z(\mathbf{x})$, that would be used as its lower bound. The process would then be repeated for each of $\mu_i(\mathbf{x})$, $i = 1, \ldots, m_1$.

2.6.2.2 Linear Programming with Fuzzy Left-Hand Sides

Rommelfanger reports that the left-hand side of a fuzzy constraint

$$\tilde{A}_{i1}x_1 \oplus \tilde{A}_{i2}x_2 \oplus \ldots \oplus \tilde{A}_{in}x_n \tilde{\leq} \tilde{B}_i$$

can be aggregated into a fuzzy set $\tilde{A}_i(\mathbf{x})$. If all coefficients, $\tilde{A}_{ij}$, of the i-th constraint are fuzzy intervals of the same L-R-type, then the left-hand side can be consolidated to a fuzzy interval with the same reference functions. For example, coefficients of the form displayed in Figure 2.2 have a representative function $\tilde{A}_{ij} = (\underline{a}_{ij}; \overline{a}_{ij}; \underline{\alpha}^{\epsilon}_{ij}; \overline{\alpha}^{\epsilon}_{ij})^{\epsilon}$, which can be consolidated into

$$\tilde{A}_i(\mathbf{x}) = \left(\underline{a}_i(\mathbf{x}), \overline{a}_i(\mathbf{x}); \underline{\alpha}^{\epsilon}_i(\mathbf{x}), \overline{\alpha}^{\epsilon}_i(\mathbf{x})\right)^{\epsilon}$$

with

$$\underline{a}_i(\mathbf{x}) = \sum_{j=1}^{n} \underline{a}_{ij}x_j, \quad \overline{a}_i(\mathbf{x}) = \sum_{j=1}^{n} \overline{a}_{ij}x_j, \quad \underline{\alpha}^{\epsilon}_i(\mathbf{x}) = \sum_{j=1}^{n} \underline{\alpha}^{\epsilon}_{ij}x_j, \quad \overline{\alpha}^{\epsilon}_i(\mathbf{x}) = \sum_{j=1}^{n} \overline{\alpha}^{\epsilon}_{ij}x_j$$

The fuzzy linear program is now expressed as a comparison of two fuzzy sets

$$\tilde{A}_{i1}(\mathbf{x}) \tilde{\leq} \tilde{B}_i$$

And this raises the question of how the inequality relation in fuzzy constraints should be interpreted. Although many concepts have been proposed, we will just give one, due to Rommelfanger, as an illustration

$$\tilde{A}_{i1}(\mathbf{x}) \tilde{\leq}_R \tilde{B}_i \Leftrightarrow \begin{cases} \sum_{j=1}^{n} (\overline{a}_{ij} + \overline{\alpha}^{\epsilon}_{ij})x_j \leq b_i + \beta^{\epsilon}_j, \\ \mu_i(\mathbf{x}) = \mu_i(\overline{a}_i(\mathbf{x})) \rightarrow \text{ Max} \end{cases}$$

where $\mu_i(\mathbf{x})$ is defined according to (2.15).

Rommelfanger refers to the first term as the 'pessimistic index', and the second term as the fuzzy goal. Rommelfanger suggests that this interpretation has some advantages over other proposals given in the literature. In particular, the inequality relation, $\tilde{\leq}_R$, coincides with the usual interpretation of the inequality relation in soft constraints (2.16), and it corresponds to the usual $\leq$ relation if the constraint is deterministic. Thus, Rommelfanger's interpretation converts the fuzzy constraints into a hard constraint and an objective, in the same way as is done with soft constraints.

2.6.2.3 Fuzzy Objective Functions

Finally, the objective function coefficients may also be fuzzy:

$$\tilde{Z}(\mathbf{x}) = \tilde{C}_1 x_1 \oplus \ldots \oplus \tilde{C}_n x_n \tag{2.18}$$

Rommelfanger suggests that this should be interpreted as a multiobjective function. In the simple case, in which all of the coefficients have been expressed as fuzzy intervals of the form $\tilde{C}_j = (\underline{c}_j; \overline{c}_j; \underline{\gamma}_j^{\epsilon}; \overline{\gamma}_j^{\epsilon})^{\epsilon}$, $\tilde{Z}(\mathbf{x})$ can be written as

$$\tilde{Z}_i(\mathbf{x}) = \Big(\underline{c}(\mathbf{x}),\, \overline{c}(\mathbf{x});\, \underline{\gamma}^{\epsilon}(\mathbf{x});\, \overline{\gamma}^{\epsilon}(\mathbf{x})\Big)^{\epsilon}$$

with

$$\underline{c}(\mathbf{x}) = \sum_{j=1}^{n} \underline{c}_j x_j, \quad \overline{c}(\mathbf{x}) = \sum_{j=1}^{n} \overline{c}_j x_j, \quad \underline{\gamma}^{\epsilon}(\mathbf{x}) = \sum_{j=1}^{n} \underline{\gamma}_j^{\epsilon} x_j, \quad \overline{\gamma}^{\epsilon}(\mathbf{x}) = \sum_{j=1}^{n} \overline{\gamma}_j^{\epsilon} x_j$$

in the same manner as was used for fuzzy left-hand sides.

The fuzzy objective function (2.18) is said to imply the four goals

Max	$\underline{c}(\mathbf{x}) - \underline{\gamma}^{\epsilon}(\mathbf{x})$	the 'worst case' scenario,
Max	$\underline{c}(\mathbf{x})$	the scenario formed from the smallest values with $\mu_j(c_j) = 1$,
Max	$\overline{c}(\mathbf{x})$	the scenario formed from the largest values with $\mu_j(c_j) = 1$,
Max	$\overline{c}(\mathbf{x}) + \overline{\gamma}^{\epsilon}(\mathbf{x})$	the 'best case' scenario,

that is, maximise for four combinations of representative settings of the fuzzy objective function coefficients. The question now is, how should a compromise solution for these four objectives be determined? Various approaches have been suggested, including the replacement of these fuzzy objectives with linear membership functions. For example, Tanaka, Ichihashi & Asai (1984) obtain a compromise solution by replacing the fuzzy objective with the crisp 'compromise objective':

$$z(\mathbf{x}) = \frac{1}{6} \sum_{j=1}^{n} \Big(2\underline{c}_j + 2\overline{c}_j + \underline{\gamma}_j + \overline{\gamma}_j\Big) x_j$$

Rommelfanger suggests the use of a fuzzy aspiration level, which can be described as

$$\tilde{N} = (n; v^{\epsilon}; 0)^{\epsilon}$$

and this forms the basis of his FULPAL procedure. In FULPAL, a compromise solution to a fuzzy linear program with fuzzy right-hand sides, and fuzzy constraint and objective function coefficients can be found by solving a crisp linear program.

2.6.3 Interval Objective Coefficients

Ishibuchi & Tanaka (1990) describe a solution approach to linear programs in which the objective function coefficients are expressed as intervals, and the constraints are deterministic. For any feasible solution, $\mathbf{x}$, the objective function value, $Z(\mathbf{x})$, is an interval. A conservative approach is to use the left limit of each objective function coefficient, and to solve the resulting "worst case" problem. Ishibuchi and Tanaka propose a bicriterion approach, in which the decision maker's preference between two interval objective function values is defined by comparing two attributes of the objective function intervals. Interval $Z(\mathbf{x})$ is preferred to interval $Z(\mathbf{y})$, if $Z(\mathbf{x})_L \geq Z(\mathbf{y})_L$ and $Z(\mathbf{x})_R \geq Z(\mathbf{x})_R$, where the subscripts L and R refer to the left and right limits of the intervals. This order relation is referred to as the *LR* relation. Similarly, interval $Z(\mathbf{x})$ is preferred to interval $Z(\mathbf{y})$, if $Z(\mathbf{x})_C \geq Z(\mathbf{y})_C$ and $Z(\mathbf{x})_W \leq Z(\mathbf{y})_W$, where the subscripts C and W refer to the centre and width of the intervals. This order relation is referred to as the *CW* relation.

The authors note that the two order relations are never in conflict. That is, there is no pair of intervals such that one interval is preferred according to the *LR* order relation, and the other interval by the *CW* order relation. They also note that these are partial order relations, and that the decision maker's preference between many pairs of intervals cannot be determined by either order relation.

A combination order relation, referred to as *LC*, is then derived, which only holds if *LR* or *CW* holds. The *LC* order relation compares the left limits and the centres of the intervals. Using this order relation, the interval objective function is reformulated as the bicriterion objective function:

$$\text{maximise} \quad \left\{ \begin{array}{l} Z_L = c_{1L}\mathbf{x}_1 + \cdots + c_{nL}\mathbf{x}_n \\ Z_C = c_{1C}\mathbf{x}_1 + \cdots + c_{nC}\mathbf{x}_n \end{array} \right\} \tag{2.19}$$

where: c_{jL} = the lower limit of the interval of coefficient j

c_{jC} = the centre of the interval of coefficient j

The two competing objectives are to maximise the objective function under the worst case scenario, and to maximise the scenario formed by combining the central values of the interval objective function coefficients. This second scenario will be the average case if the central values of the coefficients are their expected values. The idea is to find the nondominated extreme points for the decision maker to choose among.

Chanas & Kuchta (1996) extend this idea to a more general form, that can accommodate a family of interval preference relations, of which the relations of Ishibuchi & Tanaka (1990) are a special case. Again, the interval objective function is reformulated to form a bicriterion function, and the problem solved to find a set of nondominated solutions.

2.6.3.1 A Mini-Max Regret Approach

Inuiguchi & Sakawa (1995) propose a method for finding the mini-max regret solution to a linear problem with an interval objective function and deterministic constraints. The authors show that when calculating the regret for any feasible solution $\mathbf{x}$ with respect to another feasible solution $\mathbf{y}$, only the upper and lower bounds of the intervals of the objective function coefficients need be considered. They also show that the mini-max regret solution need not be an extreme point of the feasible region, although it will be on the boundary of the feasible set. They then show that only the extreme solutions to the problem need be considered in order to find the maximum regret of any feasible solution.

They develop an iterative algorithm that can find the mini-max regret solution, once the extreme points have been found. Although the authors do not appear to recognise the fact, their method employs a well established MOLP technique that uses the Tchebycheff metric to evaluate the quality of a candidate solution (see Steuer 1986). They also gloss over the requirement that all of the extreme solutions must be found as the first step in their approach. However, once they have the extreme solutions their method should find the mini-max regret solution in a finite number of iterations. At worst this will be N^2, where N is the number of objective coefficients with interval data, and is also the number of scenarios that would be formed by combining all of the upper and lower bounds of the intervals. This algorithm requires that the following LP be solved at each iteration:

$$\begin{array}{lll} \text{minimise} & r & \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} & \\ & r \geq Z_j^* - \mathbf{c}_j\mathbf{x} \qquad j = 1, 2, \ldots, k & (2.20) \\ & \mathbf{x} \geq 0 & \end{array}$$

where: r = the maximum regret

Z_j^* = the optimal solution for scenario j

k = the number of scenarios so far considered

It would seem that a much more straightforward approach would be to calculate the optimal objective value for each scenario from the extreme points, and then solve the mini-max problem directly:

$$\begin{aligned} &\text{minimise} && r \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b} \\ & && r \geq Z_s^* - \mathbf{c}_s\mathbf{x} \qquad \forall\ s \in S \\ & && \mathbf{x} \geq 0 \end{aligned} \tag{2.21}$$

where: r = the maximum regret

Z_s^* = the optimal solution for scenario s

S = the set of scenarios

This is, in fact, the LP that will be solved by the proposed algorithm if it reaches the limit of N^2 iterations.

Once the extreme points have been found, it is a simple matter to calculate all of the scenario optimal solutions immediately, rather than incrementally at each iteration of the algorithm. Although there will be a large number of additional constraints for a problem with a large number of interval objective coefficients, many of them can be expected to be redundant. It really seems that little is to be gained by using the proposed iterative approach, which solves an LP at each iteration, rather than solving the limiting problem once.

The alternative formulation (2.21) was tried on the example problem given in the paper, and the same solution was found. However, the most serious drawback of this approach, whether formulation (2.21), or (2.20), is used is the need to find all of the noninferior extreme points. As reported by Steuer (1986), even moderately sized multiobjective optimisation problems have extemely large numbers of noninferior extreme points, which means that this approach will be computationally demanding.

2.6.4 Grey Linear and Integer Programming

Grey systems theory was developed by Dr J. Deng in the 1980's to deal with the problem of uncertainties in systems analysis (see, for example, Julong 1987). The idea is that uncertain parameters can be expressed as *grey numbers*, that is, by specifying an interval in which the true value must lie. Interval, or grey linear programming has evolved from this idea as a method for applying grey systems analysis to decision making under uncertainty. The output from a grey program is a grey number for each decision variable, and a grey number for the objective function value. Huang & Moore (1993) present an approach in which both the objective function coefficients, and the constraint parameters may be grey numbers. Huang, Baetz & Patry (1995) extend this idea to include binary variables, thus enabling grey programming to be applied to facility expansion problems.

In Huang & Moore (1993) and Huang et al. (1995), the upper and lower limits of the intervals are combined to produce scenarios that are optimised to obtain the grey (interval) values for the objective function and the decision variables.

The choice of scenarios is based on the observation that the objective function will be at its maximum value when the coefficients are at their upper bounds. Similarly, it will be at its minimum value when the coefficients are at their lower bounds. The objective function of the first scenario is called the f^+ objective, and of the second, the f^- objective.

$$f^+ = \sum_{j=1}^{J} c_j^+ x_j^+ + \sum_{k=1}^{K} c_k^+ x_k^-$$

$$f^- = \sum_{j=1}^{J} c_j^- x_j^- + \sum_{k=1}^{K} c_k^- x_k^+$$

where: $c_j \geq 0$ for j = 1 ... J

$c_k < 0$ for k = 1 ... K

c_j^+ , c_k^+ denote the upper bounds

c_j^- , c_k^- denote the lower bounds

$[x_j^-, x_j^+]$ includes the optimal value of x_j for all scenarios

$[x_k^-, x_k^+]$ includes the optimal value of x_k for all scenarios

The values of the f^+ and f^- objectives define the interval for the objective

function. For convenience we will assume a maximising problem, although the logic also applies to a minimising problem, but less intuitively. If the **A** matrix also contains grey parameters, then different constraint sets are defined for each of the two objective functions. For the f^+ objective, the nonnegative elements of the **A** matrix are chosen to be at their lower bounds for the x_j variables, and at their upper bounds for the x_k variables, whereas the negative elements of the **A** matrix are chosen to be at their upper bounds for the x_j variables, and at their lower bounds for the x_k variables. (All constraints are required to be of the form $A\mathbf{x} <= \mathbf{b}$.) For the f^- objective the opposite arrangement is used. Similar logic is used to determine which of the bounds of the grey right hand side parameters should be chosen. The constraints are then divided through by these values. The elements of **A** corresponding to the x_j variables are divided by the lower bound of the right hand side parameter, while the elements of **A** corresponding to the x_k variables are divided by the upper bound of the right hand side parameter. Thus the constraints used to find the grey solutions are unlikely to correspond to any of the possible scenarios of the original problem, because a right hand side parameter cannot have two values simultaneously under one scenario.

The authors state that this formulation "can lead to grey solutions with good quality". Whereas the use of best and worst case scenarios "are unable to generate grey solutions with good quality" (Huang et al. 1995). Although they do not define what they mean by *good quality*, it appears that they want a formulation for which, not only is $x_j^+ \geq x_j^- \ \forall \ j \in J$ and $x_k^+ \geq x_k^- \ \forall \ k \in K$, but the intervals $[x_j^-, x_j^+]$ and $[x_k^-, x_k^+]$ are small. However, the proof given on page 600 of the article to show that their formulation does provide better quality solutions is flawed. They fail to consider problems in which a column of the **A** matrix has both positive and negative terms. In such problems it is not possible to ensure that $x_j^+ \geq x_j^- \ \forall \ j \in J$ and $x_k^+ \geq x_k^- \ \forall \ k \in K$.

The authors appear to be forcing the scenario problem to conform to the objective of grey integer programming (to generate grey decision variables), and they have lost sight of the decision maker's need to cover all possibilities, or at least to have information about the possibilities that are covered. Because mechanical rules are used to select the combinations of parameters which form the two "scenarios", these scenarios cannot be relied upon to be consistent, or credible. While this may not be an objective of the authors, it is unlikely to give the decision maker confidence

in the results.

In using this formulation, the authors are either assuming that the optimal values of all of the decision variables, for all scenarios, lie within the intervals generated, or that optimal values outside the intervals are of no interest to the decision maker (since the formulation doesn't report them). The first assumption is clearly not tenable, since scenarios consisting of a mix of high and low values from the intervals for the objective coefficients could easily lead to tradeoffs which generate more extreme values in some variables than the two "scenarios" used to generate the grey solutions. The second assumption is, at best, questionable.

An unfortunate consequence of their decision to not solve the worst case scenario is that problems in which the worst case scenario is infeasible are not identified. And, in fact, this is the case in the example given in Huang et al. (1995). In this example the worst case scenario is infeasible, but this insight is not reported.

An initially appealing aspect of grey integer programming is that some of the variables will be found to have the same value under both the f^+ and the f^- formulations, which the authors interpret to mean that they will have these values under all scenarios. This follows from their assumption that, for all scenarios, the optimal values for the variables will lie within the intervals generated, but this assumption is not generally supportable.

The grey integer formulation does not model the resolution of uncertainty over time, but solves the limiting scenarios assuming full and immediate resolution of the uncertainty. There is no attempt to impose nonanticipativity restrictions. This compounds a major drawback of this method: the difficulty the decision maker will have in forming a decision to implement from the intervals found for the variables. Given a solution of grey variables, the decision maker is unable to answer questions of the form: "If we implement this decision at stage one, and the uncertain parameters take these values, what recourse will be required at stage two?". This shortcoming is illustrated in the example, in which the binary variables that describe the type of expansion needed at the start of stage one are conditioned on the volumes of waste generated during stage one (and the later stages), but these volumes are uncertain.

The idea that uncertainties might be expressed as intervals, without probability distributions, and then communicated directly to the solution process is an appealing one. However, as it stands, the reported application of the method to multi-stage decision making under uncertainty is flawed. The proofs used to support the theory

are inadequate, and in at least one case, incorrect, and the authors do not discuss how the output should be interpreted to produce implementable decisions.

2.6.5 Partial Probability Information

Parkan (1994) proposes an approach to decision making under uncertainty, in which the probability of an uncertain event is expressed as a value range. The method is designed to identify efficient strategies from a set of candidate strategies. In other words, it is a method for analysing an available payoff table, rather than a method for finding solutions to the original problem. The expected opportunity cost is calculated for each strategy under each state of nature, and an LP formed to minimise the maximum expected opportunity cost over the available strategies. The decision variables are the probabilities of the states of nature, and they are constrained to be within their specified value ranges. The LP is solved for one of the strategies. If the strategy is not efficient, the strategy corresponding to a binding constraint is selected and the problem solved again. The process is repeated until all of the efficient strategies have been identified. The paper includes an example problem with two states of nature, with the probability of one of them specified to be in the range 0.6 to 0.7. Two states of nature are chosen because the objective and probability spaces are two dimensional, which facilitates graphical representation. It appears that the method is used as a teaching tool, and Parkan does not mention any other implementations. Nor does he discuss its extension to more than two states of nature, and thus to higher dimensioned objective and probability spaces.

2.7 Extensions and Implementations

2.7.1 Risk Limitation Constraints

Eppen, Martin & Schrage (1989) propose an implementation of the stochastic linear program in which constraints are appended to the problem to limit the downside risk. The idea is discussed in the context of a capacity expansion planning exercise that was carried out at General Motors. The problem was formulated with five periods, with three possible outcomes at each period, making a total of 243 scenarios. The probabilities of the three outcomes were assumed to be 0.1, 0.3 and 0.6 for all periods. The objective was to maximise the expected profit, subject to restrictions on the

expected downside risk. The problem was formulated as:

$$
\begin{aligned}
\text{maximise} \quad & \sum_{s\in S} p_s \sum_{t\in T} \mathbf{c}_s^t \mathbf{x}_s^t \\
\text{subject to} \quad & A_s^t \mathbf{x}_s^t = \mathbf{b}_s^t && \forall\, s \in S \quad t \in T \\
& Z_s \geq \tilde{z} - \sum_{t\in T} \mathbf{c}_s^t \mathbf{x}_s^t && \forall\, s \in S \\
& Z = \sum_{s\in S} p_s Z_s \\
& Z \leq \phi Z_0 \\
& Z_s \geq 0,\ \mathbf{x}_s^t \geq 0 && \forall\, s \in S \quad t \in T
\end{aligned}
\tag{2.22}
$$

where: $\mathbf{x}_s^t$ = decision variables under scenario s in period t, some of which are integer

$\tilde{z}$ = the objective value below which downside risk takes on positive values

Z_s = the amount by which the profit under scenario s falls below $\tilde{z}$

Z = the expected downside risk

Z_0 = the expected downside risk without risk limitation constraints

ϕ = a parameter between 0 and 1 used to tighten the risk limitation constraint

The problem is formulated as a preemptive goal program, in which the goal of limiting the expected downside risk is given the highest priority, while maximising the expected profit is the secondary goal. In the example given, $\tilde{z}$ was set to zero, that is, only losses were included in the downside risk. When the problem was solved without the risk limitation constraints, the histogram of profits was bimodal with an expected profit of 141 and an expected downside risk of 8.37. When the expected downside risk was limited to ≤ 7, the histogram of the profit became unimodal, with an expected profit of 132, and an expected downside risk of 0.

The authors compared the solutions that this formulation identified with those found using scenario analysis. They formed three scenarios by assuming that one of the outcomes occurred with a probability of 1 at every stage. When the outcome with probability of 0.6 was used, the solution was the same as that found without the risk limitation constraints, showing that this high probability outcome was driving

the problem. The other two scenarios produced solutions with expected downside risks of 0, but an expected profit of 124. Thus the authors found that by explicitly modelling the uncertainty they were able to identify better solutions than could be found by solving deterministic problems.

The paper does not discuss how the probabilities of the outcomes was arrived at, nor the effect on the solutions of changing these probabilities.

2.7.2 Importance Sampling

In problems with several uncertain parameters, each with several possible outcomes, the number of scenarios at each stage becomes extremely large. Suppose that a two-stage problem included 10 uncertain parameters, each with 5 possible outcomes. The number of scenarios is $5^{10} \approx 10^7$, each of which corresponds to a subproblem to be solved each time a cut is generated. In a multi-stage problem this is repeated at every stage. In response to the impossibility of solving such problems exactly sampling techniques have been developed. These techniques generate cuts at each stage by solving a small sample of the subproblems. The method requires that the uncertain parameters be expressed as discrete, independent random variables, so that any combination of outcomes from the distributions is a valid scenario. Dantzig & Glynn (1990) propose a scheme for sampling from the scenarios which they call *importance sampling.* This work has been further developed by Dantzig & Infanger (1993), see also (Infanger 1992, Infanger 1994). The sampling scheme is a modification of Monte Carlo sampling. Random variables that have large impacts on the value of the second stage objective value are sampled more frequently than those with lower impacts. The relative importance of each random variable is recalculated each time a new stage one decision vector is found. Dantzig & Glynn report that "the size of sample required to obtain the same size interval with the same degree of confidence of covering the true minimum value was 1/20,000 smaller using 'importance' sampling than would have been the case using 'naive' sampling."

Infanger (1994) reports experiments with three test problems in which importance sampling, using small sample sizes, finds solutions close to the true stochastic optimum. The test problems are sufficiently small for the full deterministic equivalent problem to be solved, so that the solutions found using importance sampling can be compared to the true stochastic optimal solutions. However, as discussed in

Chapter 6, we have experimented with one of these problems (APL1P) and found that the objective function is very flat, and a large range of solutions are almost optimal. The expected-value solution is also almost optimal, and the value of the stochastic solution is very close to zero. Because a small VSS indicates that the uncertainties in the problem do not have a large impact on the performance of the stage one decision, it would seem that this problem does not provide a demanding test of importance sampling, and that it would be more telling to test it on a problem for which the value of the stochastic solution (VSS) is large.

Infanger (1994) also reports experiments that were carried out on five large scale stochastic problems with deterministic equivalent problems that were far too large to solve. The expected-value problem is solved to provide a starting point for the stochastic solution. It is also tested under the full set of scenarios. The optimal objective value reported by the expected-value problem, and the optimal objective value found by testing it under all scenarios, provide upper and lower bounds on the true stochastic objective function value. For two of the problems these bounds are only 3% apart, while the bounds are 5%, 8% and 36% apart for the other problems. In all cases importance sampling found solutions with very tight (95%) confidence intervals. The improvement in the bounds for the last problem are certainly worthwhile. However, because the values of future parameters cannot be predicted accurately, it would seem that bounds that are less than 10% apart could well be considered to be quite tight enough. This is not to suggest that importance sampling has no value, but it is striking how many of the problems discussed in this book had very small values of stochastic solution. This suggests that it is well worthwhile starting the analysis of a large stochastic problem by finding the expected-value solution, and then using the approach proposed in Birge (1982) to see if there is any point in solving the full stochastic problem.

2.7.3 Scenario Aggregation

As discussed in Section 2.3.2, scenario analysis can be used to find the optimal solution for each scenario individually, but this approach lacks a systematic method for finding a solution to implement. Rockafellar & Wets (1991) (see also Wets 1989) propose the *progressive hedging* algorithm as an iterative technique for finding a solution that is "well hedged" with respect to all of the scenarios. In contrast to

the methods based on Bender's decomposition, which decompose the problem by scenario and stage, scenario aggregation decomposes the problem by scenario only. As in scenario analysis, the problem is solved for each scenario, that is, for each complete path through the decision tree, from the root to a leaf. This produces the optimal decision vector at each stage, for each scenario, but, unless the decision vectors at a node are the same for all branches leaving the node, these decisions cannot be implemented, because the decision maker cannot predict which branch will be followed. Rockafellar and Wets use the term *implementable* to describe decisions that are the same for all branches at a node. The requirement that all decisions be implementable is referred to as the *nonanticipativity requirement*, that is, the decision maker does not have to anticipate the future in order to determine which decision should be implemented.

In the progressive hedging algorithm the decisions at each node that are different for different branches leaving the node (and thus not implementable) are used to calculate penalty terms which are used to augment the objective function. The problem is solved again to find new scenario optimal decisions, and new penalty terms calculated. Thus the nonanticipativity requirement is progressively imposed. The calculation of the penalty term includes scenario weighting terms, that can be interpreted as scenario probabilities. Thus, as with stochastic optimisation, scenario aggregation maximises the weighted sum of the scenario objective function values. If the weighting terms are scenario probabilities, then scenario aggregation maximises the expected objective function value. Rockafellar & Wets show that, in the convex case, the algorithm will converge in a finite number of iterations.

The progressive hedging algorithm relies on separability (with respect to the scenarios) of all problem elements except the nonanticipativity constraint. Robinson (1991) extends the technique to include non-separable convex constraints, and uses a portfolio optimisation problem of the Markowitz type as an illustration. Jörnsten (1992) applies scenario aggregation to a stochastic mixed integer planning problem, and Jönsson, Jörnsten & Silver (1993) apply it to a two-stage inventory problem. Chun & Robinson (1995) propose *bundle decomposition* as an alternative to the progressive hedging algorithm for solving the scenario aggregation problem. They apply their method to a set of test problems and deduce that the bundle decomposition method is superior to progressive hedging for loosely coupled problems. A problem

is *loosely coupled* if the number of decision variables (those to which the nonanticipativity restriction applies) is small compared to the total number of scenario-specific variables. In the most extreme of their examples, there are 36 decision variables and 619 scenario-specific variables.

Berland & Haugen (1996) propose a variation of the progressive hedging algorithm in which the "scenarios" to be aggregated by the hedging algorithm can themselves include uncertainties. They illustrate this idea with an event tree with three stages, each with binary chance nodes, leading to eight scenarios. They gather the four scenarios on one of the branches at the first chance node into one "scenario" and the four scenarios on the other branch into a second "scenario". These two "scenarios" are themselves stochastic problems, which are solved using stochastic dynamic programming. The progressive hedging algorithm is used to blend the two solutions into a single hedged solution. This is referred to as a *hybrid* approach, being a mix of the progressive hedging algorithm (PHA), and stochastic dynamic programming (SDP). When all of the scenarios in the event tree are gathered into one "scenario" the problem is purely SDP. When all of the scenarios are kept separate, the problem is purely PHA, and arrangements in between are hybrid.

They illustrate this idea with a stochastic, nonlinear control problem from macroeconomics. They experiment with different numbers of parallel processors to solve the scenario subproblems, and by varying the mix from pure SDP through to pure PHA. They also experiment with settings for the penalty parameter, ρ, used in the augmentation of the objective function by scenario aggregation. They conclude that a pure SDP algorithm is faster than the hybrid PHA-SDP algorithm, but point out that the test problem has a two dimensioned state space. They suggest that problems with larger state spaces could be intractable for a pure SDP algorithm, but could still be solved using the hybrid approach.

The experiments with the number of parallel processors shows that the processors are utilised most efficiently when the number of processors matches the number of scenarios. For example, when they arranged the test problem to have 81 scenarios, the master processor, which farms out work to the slave processors, and the slave processors, ran at an efficiency of 0.78 when there were 82 processors, but at 0.56 when there were 70 processors. This suggests that, if the number of processors is fixed, the hybrid PHA-SDA can be used to match the number of scenarios to the number of processors.

2.7.4 Scenario Optimisation

Dembo (1991) proposes a formulation for solving stochastic problems, that he calls *scenario optimisation.* This approach assumes that the uncertainties are represented by a set of scenarios emanating from a single chance node. The problem is solved for each scenario, and then a *tracking model* is used to find a single, feasible policy. The scenario subproblems are expressed as:

$$\begin{array}{lll} \text{minimize} & v_s = \mathbf{c}_s\mathbf{x}_s & \\ \text{subject to} & A_s\mathbf{x}_s = \mathbf{b}_s & \text{scenario specific constraints} \\ & A_d\mathbf{x}_s = \mathbf{b}_d & \text{deterministic constraints} \\ & \mathbf{x}_s \geq 0 & \end{array} \tag{2.23}$$

where: v_s is the optimal objective value for scenario s

$\mathbf{x}_s$ is the optimal solution for scenario s

the scenario specific constraints contain the uncertain parameters

the deterministic constraints are the same for all scenarios

A tracking (or coordination) problem is defined to find a "reasonable" solution to the overall stochastic problem. The scenario constraints are moved into the objective function as penalty terms, and deviations from scenario optimality are also penalised. The tracking problem has the form:

$$\begin{array}{ll} \text{minimize} & \sum_s p_s \|\mathbf{c}_s\mathbf{x} - v_s\|^p + \sum_s p_s \|A_s\mathbf{x} - \mathbf{b}_s\|^p \\ \text{subject to} & A_d\mathbf{x} = \mathbf{b}_d \\ & \mathbf{x} \geq 0 \end{array} \tag{2.24}$$

where: $\|\cdot\|^p$ denotes an L_p-norm, and is chosen to suit the problem

Dembo summarises the procedure of the scenario optimisation approach as:

Stage 1: Find a solution to the (deterministic) problem under every scenario

Stage 2: Solve a coordinating or tracking model to find a single, feasible policy

The discussion moves to the use of the L_1-norm to formulate the objective function, and observes that this is equivalent to the recourse model with simple recourse:

$$\begin{array}{lll} \text{minimize} & \sum_s p_s[(w_s^+ + w_s^-) + e^T(\mathbf{y}_s^+ \quad + \mathbf{y}_s^-)] & \\ \text{subject to} & A_d\mathbf{x} = \mathbf{b}_d & \\ & A_s\mathbf{x} - \mathbf{b}_s \; -(\mathbf{y}_s^+ - \mathbf{y}_s^-) \quad = 0 & \forall\, s \in S \\ & \mathbf{c}_s\mathbf{x} - v_s \; -(w_s^+ - w_s^-) \quad = 0 & \forall\, s \in S \\ & \mathbf{x},\ \mathbf{y}_s,\ w_s \geq 0 & \end{array} \tag{2.25}$$

However, this is a special case of simple recourse in which the second stage term of the objective function is usually given as:

$$\sum_s p_s(\mathbf{q}_s^+\mathbf{y}_s^+ + \mathbf{q}_s^-\mathbf{y}_s^-)$$

(see, for example, Wets 1983, Kall & Wallace 1994). In Dembo's formulation the objective function coefficients are set to 1 for all feasibility deviations, which implies an assumption that the deviations are directly comparable between constraints, and that feasibility deviations are directly comparable to deviations from optimality. Clearly this assumption will encounter major difficulties with respect to units of measure, except in special cases. In this paper (1991), and in a subsequent paper (Dembo 1993), a portfolio immunization problem is discussed in which the objective function and the constraints that can be violated are both in net present value units:

$$\begin{array}{lll} \text{minimise} & v_s = \mathbf{c}_s\mathbf{x}_s & \\ \text{subject to} & \mathbf{l} \leq \quad \mathbf{x}_s \quad \leq \mathbf{u} & \\ & PV_s(\mathbf{x}_s) \geq PV_{sT} & \\ & \mathbf{c}_s\mathbf{x}_s \quad \leq C & \end{array} \tag{2.26}$$

where: $\mathbf{x}_s$ = the selected portfolio under scenario s

$PV_s(\mathbf{x}_s)$ = the present value of portfolio $\mathbf{x}_s$ under discount scenario s

PV_{sT} = the net present value of the target portfolio under discount scenario s

C = the maximum investment amount available

The objective is to find a portfolio, $\mathbf{x}$, that will be able to cover a given portfolio of

liabilities under a number of different discount scenarios. This problem arises, for example, in the context of pension fund management where the fund must be invested in such a way that it can meet the fund's projected liabilities. Problem (2.26) is solved to find the optimal investment portfolio for each scenario, and a tracking problem is used to blend these scenario optimal portfolios into a single portfolio that performs satisfactorily under all scenarios. The suggested tracking problem has the form:

$$\begin{aligned} \text{minimise} \quad & \sum_{s\in S} p_s\Big[\big(\mathbf{c}_s\mathbf{x} - v_s\big)^2 + \big(PV_s(\mathbf{x}) - PV_{sT}\big)^2\Big] \\ \text{subject to} \quad & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \mathbf{c}_s\mathbf{x} \leq C \qquad \forall\, s \in S \end{aligned} \tag{2.27}$$

The budget constraint is retained as a hard constraint, which must be satisfied under all scenarios. This must be the case, because the investment portfolio has to be decided before the scenarios are known. The formulation does not include recourse variables, so the model does not include any opportunities the decision maker may have to rebalance the portfolio later. Although use of this model to find immunization portfolios may well be superior to the deterministic model traditionally used (as stated in the 1991 paper), it is not apparent why Dembo has chosen such a minimal representation of the decision process.

The second example given in the 1991 paper, (taken from Dembo, Chiarri, Martin & Paradinas 1990) is the multi-period hydro scheduling problem:

$$\text{maximize} \quad \sum_{t=1}^{T}\sum_{j=1}^{J} B_{tj}(V_{t-1,j}, V_{tj}, R_{tj}) \tag{2.28 a}$$

$$\text{subject to} \quad V_{tj} - V_{t-1,j} - \sum_{k\in K_j}(R_{tk} + S_{tk}) + R_{tj} + S_{tj} = I_{tj}\,, \tag{2.28 b}$$

$$\underline{R}_{tj} \leq R_{tj} \leq \overline{R}_{tj}\,, \tag{2.28 c}$$

$$\underline{V}_{tj} \leq V_{tj} \leq \overline{V}_{tj}\,, \tag{2.28 d}$$

$$S_{tj} \geq 0 \qquad \forall\ t = 1,\ldots,T \text{ and } j = 1,\ldots,J$$

where: $B(\cdot)$ = a stochastic nonlinear function measuring the benefit of hydro vs. thermal generation

V_{tj} = the volume of reservoir j at the end of period t; $\overline{V}$ and $\underline{V}$ are given upper and lower limits on V

I_{tj} = the net (stochastic) inflow to reservoir j in period t

R_{tj} = the release (for generation) from reservoir j in period t; $\overline{R}$ and $\underline{R}$ are given upper and lower limits on R

S_{tj} = the amount spilt from reservoir j in period t

K_j = the set of reservoirs immediately upstream from reservoir j

The tracking problem is not given for this example (nor in Dembo et al. 1990), but, conforming to the formulations discussed in the paper, it would be of the form:

$$\text{minimise} \quad \sum_{s\in S} p_s \|B(\cdot)_s - B(\cdot)^*_s\|^p \tag{2.29 a}$$

$$+ \sum_{s\in S} p_s \|\alpha^{+s}_{tj} + \alpha^{-s}_{tj} + \beta^{+s}_{tj} + \beta^{-s}_{tj} + \gamma^{+s}_{tj} + \gamma^{-s}_{tj}\|^p \tag{2.29 b}$$

subject to

$$V^s_{tj} - V^s_{t-1,j} - \sum_{k\in K_j} (R^s_{tk} + S^s_{tk}) + R^s_{tj} + S^s_{tj} = I^s_{tj} - \alpha^{+s}_{tj} + \alpha^{-s}_{tj} \qquad \forall\, s \in S\,, \tag{2.29 c}$$

$$\underline{R}_{tj} \le R^s_{tj} - \beta^{+s}_{tj} + \beta^{-s}_{tj} \le \overline{R}_{tj} \qquad \forall\, s \in S\,, \tag{2.29 d}$$

$$\underline{V}_{tj} \le V^s_{tj} - \gamma^{+s}_{tj} + \gamma^{-s}_{tj} \le \overline{V}_{tj} \qquad \forall\, s \in S\,, \tag{2.29 e}$$

$$S^s_{tj},\ \alpha^{+s}_{tj},\ \alpha^{-s}_{tj},\ \beta^{+s}_{tj},\ \beta^{-s}_{tj},\ \gamma^{+s}_{tj},\ \gamma^{-s}_{tj} \ge 0\,,$$

$$\text{for all } t = 1,\ldots,T\,;\ j = 1,\ldots,J$$

where: term (2.29 a) penalises deviations from the scenario optima

term (2.29 b) penalises deviations from feasibility

Presumably, the term (2.29 a) is in dollars, whereas the α^s_{tj} and the β^s_{tj} variables will be in flow units, and the γ^s_{tj} variables will be in units of volume. Clearly the units must be reconciled when gathering the terms into the objective function, but Dembo does not discuss this issue. Another fundamental difficulty, which is not discussed, is how to handle deviations in stochastic constraints which model physical laws. Constraint (2.28 b) is such a constraint. When it is formulated as constraint (2.29 c) the solver can "create", or "destroy" water to minimise the objective function of the tracking problem. If the trade-offs in the objective function between terms

(2.29 a) and (2.29 b) are such that the objective value can be decreased by giving α_{tj}^{+s} or α_{tj}^{-s} a non-zero value, then the solver will do so, and the original flow balance constraint (2.28 b) will no longer hold, which is a nonsense. Similarly, the presence of the variables β_{tj}^{s} and γ_{tj}^{s} permit the storage and release bounds to ignored by the solver. This means that constraints that model physical laws must be retained as hard constraints, and so the objective function of this tracking problem reduces to be term (2.29 a) alone. If $p = 1$ then the problem becomes the standard stochastic optimisation problem. If $p \neq 1$ then it becomes a variance minimisation problem, similar to the robust optimisation problem proposed by Mulvey, Vanderbei & Zenios (1995) which is discussed in Section 2.7.5.

Another difficulty with Scenario Optimisation is that it models the scenarios as separate "stems", in which the uncertainty is fully resolved at a single chance node, rather than modelling the scenarios as branched trees. In stochastic programming terms this is a two-stage model, and nonanticipativity restrictions cannot be imposed after the initial stage. It also means that, after the chance node, the scenarios are deterministic, and it is widely recognised that deterministic optimisation formulations produce overly optimistic solutions which recommend extreme decisions (see, for example, Read & Boshier 1989). This means that scenario optimisation can be expected to over estimate the ability of the decision maker to recover from a decision that prepares poorly for a particular scenario. Thus, when the problem is solved again at the next stage in the rolling horizon, the decision maker may be in an extreme position, from which it will be difficult to recover. It seems likely that a formulation that includes several stages would produce less extreme initial decisions, and require less corrective action under extreme realisations of the uncertainty.

As with stochastic optimisation in general, scenario optimisation assumes that the scenario probabilities are available. All terms in the objective function are conditioned by these probabilities, and there is no discussion of how the optimal solution will change if the assumed probabilities are changed. In his 1991 paper, Dembo criticises multi-stage stochastic optimisation techniques in general, and scenario aggregation in particular, for assuming that the probabilities of uncertain events in the more distant future can be predicted well. He suggests that the current decision should not be conditioned on such unreliable parameters, but that the model should be solved periodically to adjust the policy over time. However, he does not elaborate on the implication that multi-stage models cannot also be solved periodically over

time. Neither does he explain why he excludes these unreliable parameters from the model entirely, replacing them with an assumption of certainty. It would seem that a model that includes some representation of uncertainty after the first chance node would produce better hedged solutions than the formulation proposed here, which ignores them entirely.

2.7.5 Robust Optimisation

Mulvey, Vanderbei & Zenios (1995) propose a *robust optimisation* approach for solving problems in which the uncertain data is represented by a finite number of scenarios. The authors suggest that the *reactive* approach of sensitivity analysis, which investigates the impact of data uncertainties on a model's recommendations, is inadequate, and that a *proactive* approach is needed. "That is, we need model formulations that, by design, yield solutions that are less sensitive to the model data, than classical mathematical programming formulations." They propose *robust optimisation* as an alternative to stochastic linear programming, and say that it " ... has some advantages over stochastic linear programming and is more generally applicable."

Two types of robustness are defined. A solution to problem (2.1), on page 27, that remains "close" to optimal for any of the scenarios, $\xi \in \tilde{\xi}$, is termed *solution robust.* A solution is robust with respect to feasibility if it remains "almost" feasible for any realisation of ξ. It is then termed *model robust.* Problem (2.1), is reformulated as:

$$\begin{aligned}
&\text{minimise} && \sigma(\mathbf{x}, \mathbf{y}_s) + \omega\rho(\mathbf{z}_s) && && (2.30\,\text{a})\\
&\text{subject to} && A\mathbf{x} = \mathbf{b} && && \\
& && B_s\mathbf{x} + C_s\mathbf{y}_s + z_s = \mathbf{e}_s && \forall \quad x \in S && (2.30\,\text{b})\\
& && \mathbf{x} \geq 0, \qquad \mathbf{y}_s \geq 0 && \forall \quad x \in S &&
\end{aligned}$$

where: z_s = a vector of errors that measures the infeasibilities in the recourse constraints

$\sigma(\cdot)$ = the solution robustness term, and is a function of the decision variables

$\rho(\cdot)$ = the model robustness term, and is a function of the infeasibilities

S = the set of scenarios

ω = a goal programming weight that trades off solution robustness against model robustness

For the solution robustness term, $\sigma(\cdot)$, the authors suggest the use of the mean-variance model to trade off expected return against variance in the return:

$$\sigma(\cdot) = \sum_{s \in S} p_s \xi_s + \lambda \sum_{s \in S} p_s \left(\xi_s - \sum_{s' \in S} p'_s \xi_{s'} \right)^2$$

where: ξ_s = the objective function value under scenario s

λ = the trade-off weight between minimising the expected value and its variance

For the model robustness term, $\rho(\cdot)$, two alternative penalty functions are considered:

1. The quadratic penalty function: $\rho(\cdot) = \sum_{s \in S} p_s z_s^T z_s$ for equality constrained problems, in which both positive and negative violations of the constraints are equally undesirable.

2. The exact penalty function: $\rho(\cdot) = \sum_{s \in S} p_s \max(0, z_s)$ for inequality constraints, in which only positive violations of the constraints are of interest.

The authors point out that Dembo's scenario optimisation is a special case of robust optimisation. In scenario optimisation, the optimality term penalises deviations from the optimal objective function value for each scenario, whereas robust optimisation penalises the variability of the objective function value between the scenarios.

Among the examples given, this approach is applied to a two-stage, stochastic,

power system capacity expansion problem:

$$\text{minimise} \quad \sum_{s\in S} p_s\xi_s + \lambda \sum_{s\in S} p_s\Big(\xi_s - \sum_{s'\in S} p'_s\xi_{s'}\Big)^2 + \omega \sum_{s\in S} p_s\Big(\sum_{i\in I}(z^s_{1i})^2 + \sum_{j\in J}(z^s_{2j})^2\Big) \quad (2.31\,\text{a})$$

$$\text{subject to:} \quad x_i - \sum_{j\in J} y^s_{ij} = z^s_{1i}, \qquad \text{for all} \quad i\in I,\ s\in S \quad (2.31\,\text{b})$$

$$d^s_j - \sum_{i\in I} y^s_{ij} = z^s_{2j}, \qquad \text{for all} \quad j\in J,\ s\in S \quad (2.31\,\text{c})$$

$$x_i \geq 0,\ y^s_{ij} \geq 0 \qquad \text{for all} \quad i\in I,\ j\in J,\ s\in S$$

where: $\xi_s = \sum_{i\in I} c_i x_i + \sum_{i\in I}\sum_{j\in J} f_i y^s_{ij}$

x_i = stage one decision variables (capacity to install of station type i)

y^s_{ij} = stage two recourse decision variables (allocation of capacity to demand)

d^s_j = the uncertain demand, which is modelled as a set of demand scenarios

This formulation trades off the minimisation of three objectives: the expected cost, the variance of the cost, and violations of the constraints. The reason given for minimising the variance of the cost, is that " ... cost structures that are less volatile over time are easier to defend in front of administrative and legislative boards". The minimisation of the constraint violations is to ease the need to make arrangements with other utility companies to meet temporary shortages. Choice of the factors λ and ω determine the relative weights given to each of the objectives. The authors report solutions in which λ is set to each of 0, 0.001, 0.01, 0.1, and 1.0, while ω is varied between 0 and 1000.

Violations of the capacity constraints, (2.31 b), are given equal weight to violations of the supply constraints, (2.31 c). However, these constraint violation variables are in different units: MW and MW-hours. Also, the expected value term in the objective function is in dollar units, whereas the cost variance term is in units of (dollars)^2. The implications of mixing units in the objective function are not discussed, and neither is there any discussion of how the trade-offs between the three objectives relate to the real world situation, or of how the results should be interpreted.

This formulation implicitly places a value on reducing the need to buy in power from other utilities, and this value increases quadratically as the shortages increase. However, it would seem that explicit prices could be put on power that is bought in from outside, and the supply violation variables, z^s_{2j}, would become a source of

supply and appear as linear (or piece-wise linear) terms in the objective function. It also seems unnecessary for the cost of overcapacity to appear separately in the objective function. Overcapacity is included in the cost of expansion, **cx**, and there is no explanation of why it is included again as a penalty term, nor of why it is penalised quadratically. Overcapacity is a hedge against shortages, and it seems more appropriate to directly trade off the cost of providing spare capacity against the cost of covering shortages.

Although this paper proposes an approach for finding robust solutions to problems with uncertain data, it does not discuss the reliability of the scenario probabilities. In the examples, both the $\sigma(\cdot)$ and the $\omega\rho(\cdot)$ terms are conditioned by the scenario probabilities. As in the stochastic optimisation literature, it is assumed that the uncertainties have been summarised as a set of scenarios, and that the scenario probabilities are available. Thus, although robust optimisation is designed to be proactive in its treatment of uncertainty, there is the implicit assumption that the scenario probabilities are known with certainty. Unfortunately, there is usually a very high degree of uncertainty about the probabilities of the scenarios.

Robust optimisation reformulates the linear stochastic optimisation model as a nonlinear, goal programming model. The goal programming weights must be assigned values to reflect the decision maker's trade-off preferences between the three objectives. In practice it is extremely unlikely that the decision maker will be able to specify these preferences exactly, and the problem will have to be solved for many different combinations of the weights. In addition, the objectives are conditioned by the scenario probabilities, which will normally be highly uncertain. This uncertainty about the probabilities means that the problem should also be solved for different settings of these parameters.

Robust optimisation is designed to be an improvement on linear stochastic optimisation, but it exacts a very high computational cost, both because the problem becomes nonlinear, and because the problem will have to be solved many times. The objective function includes terms with different units, which makes it difficult to interpret the goal programming weights, and to interpret the results. It would seem that a model that keeps the problem linear, and explicitly trades off between the scenarios would be more computationally tractable, and produce results that would be easier for decision makers to understand and interpret. This is the approach developed in this thesis.

In a later paper, Dai, Carpenter & Mulvey (1997) discuss the criticism that the robust optimisation model of a problem imposes an unacceptably high computational burden compared to the equivalent linear model. They suggest that the additional effort is not excessive and that much higher quality solutions are obtained than can be found using linear models. The discussion talks of incorporating " ... a concave risk aversion function in the specification of the objectives.", but they do not discuss how such risk aversion functions can be determined. As in the (1995) paper, this paper also assumes that the scenario probabilities are given, and it does not consider what effect uncertainty about these probabilities would have on the optimal solution, or what might be done to make a robust solution insensitive to variations in the probabilities.

In a way, however, worrying about the probabilities is to miss the point. Robust optimisation is designed to produce solutions that perform well under all scenarios, and so the probability of a particular scenario occurring is less important than would be the case if the performance of the solution varied widely between scenarios. Nevertheless, the scenario probabilities are used to condition the objective function, and so consideration should be given to the fact that they are normally unreliable. Also, inclusion of scenario probabilities brings robust optimisation back to the assumption of a risk neutral decision maker who wishes to maximise (or minimise) the expected value of the objective function.

2.7.6 An Implementation of Robust Optimisation

Gutierrez & Kouvelis (1995) apply robust optimisation to the problem of developing an international supplier network that is robust with respect to changes in exchange rates. The problem is formulated as a two-stage stochastic program with fixed recourse in which the stage one variables are binary (include a supplier, or not), and the recourse variables are continuous. The decision to include a supplier in the network imposes a fixed cost. The recourse variables are the quantities obtained from the suppliers who were included in the network. It is assumed that the recourse variables can take any value above a contracted minimum, which makes the problem an uncapacitated facility location problem. The only uncertain quantities are in the objective function coefficients, and this uncertainty is represented by a set of exchange rate scenarios.

The solution method starts by finding the optimal solution for each scenario. The problem is then formulated to find the supplier network that minimises the maximum deviation from the scenario optimal solutions:

$$\underset{\mathbf{y}}{\text{Minimise}} \quad R(\mathbf{y}) = \max_{s \in S}\left\{\frac{Z_s(\mathbf{y}, \mathbf{x}^s) - Z_s^*}{Z_s^*}\right\} \tag{2.32a}$$

$$\text{Subject to} \quad x_{ij}^s \leq y_j \qquad \forall\, i \in I,\ j \in J,\ s \in S \tag{2.32b}$$

$$y_j \in \{0, 1\} \qquad \forall\, i \in I \tag{2.32c}$$

$$R(\mathbf{y}) \leq R_{\max} \tag{2.32d}$$

$$\mathbf{x}^s \geq \mathbf{0} \qquad \forall\, s \in S$$

where: $R(\mathbf{y})$ = the regret over all scenarios if decision $\mathbf{y}$ is selected

$R_{\max}$ = the maximum acceptable deviation from the scenario optimal objective values

$Z_s(\mathbf{y}, \mathbf{x}^s)$ = the objective function value under scenario s if decision $\mathbf{y}$ is selected

Z_s^* = the optimal objective function value for scenario s

x_{ij}^s = the fraction of demand of factory j supplied by supplier i under scenario s

y_i = 1 if supplier i is selected, 0 otherwise

The problem is solved to find a set of solutions that meet the robustness criterion of constraint (2.32 d), rather than a single network of suppliers. A specialised branch and bound algorithm is used to find these solutions. This algorithm is designed to do a single pass through the branch and bound tree, during which all scenarios are considered at each node, before moving to the next node. A branch of the tree is fathomed when one of the scenario solutions on that branch is found to violate constraint (2.32 d). In other words, the usual approach of using the best solution found so far as the incumbent in a search for the best solution, is replaced by setting a minimum performance level, and accepting all solutions that meet that level.

An appealing aspect of this formulation is that the scenarios do not need to be assigned probabilities, and the decision maker is not assumed to have a "best expected value" preference structure. As an aside, however, the authors show that for this problem any solution that satisfies constraint (2.32 d) for all scenarios, also

satisfies it with respect to the expected value. This is,

$$\begin{aligned} &\text{if} && R(\mathbf{y}) \leq R_{\max} && \text{in problem (2.32)} \\ &\text{then} && \left\{\frac{Z_s(\mathbf{y},\mathbf{x}^s) - E(Z)}{E(Z)}\right\} \leq R_{\max} \end{aligned}$$

where: $E(Z) =$ the expected objective function value over the scenarios

This holds for all realisations of the scenario probabilities.

This formulation generates a set of good solutions for the decision maker to choose from, rather than producing a single optimal solution, and it avoids the need to assign probabilities to the scenarios. However, it is designed for a specialised problem with a special structure that can be exploited by the solution algorithm. The uncertainty is restricted to the objective function, and the problem has only two stages. Because there are no upper bounds on the capacities of the suppliers, the optimal delivery pattern for any particular supplier network can be determined by applying a greedy algorithm. This makes the calculation of the regret for any scenario very straight forward and computationally cheap. Its extension to more general problems would be difficult, and probably computationally intractable.

2.8 Conclusions

We opened this chapter with a review of stochastic programming with recourse as originally proposed by Dantzig and Beale, and subsequently developed over the past thirty years. We then considered other approaches to modelling uncertainty and to finding solutions to problems in which some of the parameters are not known with certainty, but can be described as random variables. We then reviewed the fuzzy, grey and interval programming literature in which the uncertain parameters are described as intervals, within which the true value must lie, rather than as random variables.

Extensions and implementations of stochastic optimisation were considered next. Eppen et al. (1989) propose an implementation in which additional constraints are included to limit the downside risk. The problem remains linear under this formulation. Dembo (1991) proposes "scenario optimisation", and Mulvey, Vanderbei & Zenios (1995) propose "robust optimisation", of which Dembo's scenario optimisation is a special case. In these formulations the objective function of the stochastic

optimisation problem is augmented with nonlinear penalty terms of two types. The first penalises variability of the objective function values across the scenarios, while the second type penalises violation of the RHS's of the stochastic constraints, thus converting those constraints into soft constraints. The augmentation of the objective function with the penalty terms turns the problem into a nonlinear multiobjective optimisation problem, which increases the computational demands of solving it, and raises the question of how the relative weights for the different objectives should be determined. The intention of these formulations is to find solutions that are more consistent across the possible outcomes of the uncertainty, than are the solutions found by the traditional stochastic optimisation formulation.

Two striking insights that come from reviewing the literature are that the decision maker is assumed to be risk neutral, and that the uncertainty is assumed to be described as scenarios with probabilities. Robust optimisation proposed by Mulvey, Vanderbei & Zenios (1995) is designed to improve on the assumption of risk neutrality, but it still assumes that the scenarios are assigned probabilities, and that the terms of the objective function should be weighted by those probabilities. Eppen et al. (1989) also relax the assumption of risk neutrality, but they still assume that scenario probabilities are available. This means that, as it stands, stochastic optimisation is a suitable approach for dealing with problems that Courtney et al. (1997) would classify as level 2, but that it is inappropriate to apply it to problems that would be classified as being at level 3 or level 4. That is, to problems in which the uncertain future cannot be summarised into a set of scenarios that describe all of the possible futures. Further, although it may be possible to develop representative scenarios for level 3 problems, it is not possible to assign them probabilities. This means that one of the basic assumptions of stochastic programming, that scenario probabilities are available, is not tenable. As discussed in Chapter 1, many strategic decision making problems are at level 3 and level 4. The work described in this study is designed to address this difficulty, and to enable mathematical programming to be applied to a wider class of problems. That is, to decision making under uncertainty in which the problem is at level 3. Level 4 problems, in which 'anything could happen', are not at all amendable to formulation as mathematical programmes, and the approach proposed here is not intended to be applied to them.

Thus, this study develops an approach to decision making under uncertainty in which the assumption that the scenarios will be assigned probabilities can be

relaxed. This means that the scenarios do not have to represent all possible futures. It also means that the decision maker does not have to be risk neutral.

Chapter 3

A Brief Review of Multiobjective Optimisation

3.1 Introduction

We have called our approach to decision making under uncertainty *Noninferior Set Scenario Analysis* (or NSSA). This approach is based on techniques adapted from multiobjective optimisation, which we briefly review in this chapter in preparation for the development of the theoretical basis of NSSA in Chapter 4.

In Section 3.2 we describe the multiobjective optimisation problem, and in Section 3.3 we discuss two set generation techniques, the weighting method and the constraint method. In Section 3.4 we review an interactive technique, the Techebycheff Procedure. In Section 3.5 we consider the addition of integer variables to the multiobjective optimisation problem, and very briefly review the literature. This section prepares for the extension of our approach, in Chapter 7, to decision making under uncertainty in which some of the stage one variables are binary.

3.2 Multiobjective Optimisation Problems

The multiobjective (or multicriteria, or vector optimisation) problem arises when the decision maker needs to evaluate a decision according to two, or more, criteria. In the mathematical programming context these multiple criteria lead to multiple objective functions, all of which are to be optimised simultaneously. However, the objectives are incompatible in the sense that it is not usually possible to find a solution that

produces optimal values for all of the objectives simultaneously. Achieving optimal values for some objectives imposes the penalty of accepting suboptimal values for some of the others.

Multiobjective optimisation is discussed here because decision making under uncertainty, in which the uncertainty is represented as scenarios, can be formulated as a multiobjective problem. Each scenario can be considered to have its own objective function, and the decision maker must trade off the achievement of a good outcome under one scenario against poorer outcomes under one, or more, of the other scenarios. This idea forms the basis of the approach to decision making under uncertainty that is proposed in this thesis.

An example of a multiobjective problem with K objectives is shown below:

$$\text{maximise} \quad \left\{ \begin{array}{rl} z_1 = & \mathbf{c}_1\mathbf{x} \\ z_2 = & \mathbf{c}_2\mathbf{x} \\ \vdots & \vdots \\ z_K = & \mathbf{c}_K\mathbf{x} \end{array} \right\} \tag{3.1 a}$$

$$\text{subject to} \quad \begin{array}{r} A\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq 0 \end{array} \tag{3.1 b}$$

Let $X = \{\mathbf{x} | A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq 0\}$ be the set of solutions that are feasible in constraints (3.1 b). For each $\mathbf{x} \in X$ there is a vector of objective function values $Z(\mathbf{x}) = (z_1(\mathbf{x}), z_2(\mathbf{x}), \ldots, z_K(\mathbf{x}))$ in K-dimensional criterion space. A solution vector, $\mathbf{x}$, is said to be *strongly efficient* if it is feasible, and no other feasible solution vector, $\hat{\mathbf{x}} \in X$, exists, such that $z_k(\hat{\mathbf{x}}) \geq z_k(\mathbf{x})$ for all $k \in K$, with $z_k(\hat{\mathbf{x}}) > z_k(\mathbf{x})$ for at least one of k. For an efficient solution $\mathbf{x} \in X$, the corresponding criterion vector $Z(\mathbf{x})$ is said to be *noninferior*. The use of "efficient" to describe solutions and "noninferior" to describe criterion vectors comes from Steuer (1986), and this distinction is adopted here for its convenience. However, earlier authors (e.g. Cohon 1978, Chankong & Haimes 1983) use the term noninferior to describe both solutions and criterion vectors.

The full set of efficient solutions will be referred to as X^*, and the full set of noninferior criterion vectors as N. The mapping from X^* to N is one to one, but the reverse mapping from N to X^* may be one to many.

The idea of noninferiority can be stated in terms of trade-offs: *At a noninferior*

point in criterion space, the value of any objective cannot be improved without making the value of at least one of the other objectives worse. Multiobjective problems normally have many efficient solutions (in continuous problems an infinite number), and the decision maker must trade off between the objectives when choosing between these solutions. The multiobjective problem can be considered solved when a solution is found that produces a noninferior criterion vector acceptable to the decision maker.

The multi-criteria literature makes a distinction between weak and strong efficiency. The definition of efficiency given above is that of strong efficiency, and the approach proposed in this thesis uses strongly efficient solutions. A solution vector, $\mathbf{x}$, is said to be *weakly efficient* if it is feasible, and no other feasible solution vector, $\hat{\mathbf{x}} \in X$, exists, such that $z_k(\hat{\mathbf{x}}) > z_k(\mathbf{x})$ for all $k \in K$. In terms of trade-offs, it is possible to improve the value of one (or more) of the objective values of a weakly efficient solution while keeping the other objective values constant. In MOLP's, the criterion vectors of weakly efficient solutions lie on a facet in criterion space that is parallel to one, or more, of the axes.

The algorithms presented in this thesis ensure that all of the solutions included in the efficient set are strongly efficient. We will, therefore, refer to such solutions as being efficient, with the understanding that such solutions are, in fact, strongly efficient.

An assumption that is essential to multiobjective optimisation is that the decision maker is only interested in efficient solutions, and would never wish to choose a dominated solution. In other words, the decision maker is assumed to have a monotonically increasing (or decreasing for minimisation) utility function. That is, for a maximising problem, more is always better. An implication of this assumption is that the decision maker will not be interested in weakly efficient solutions, because at least one objective value can be improved by moving to a strongly efficient solution.

A special case occurs when there is a feasible solution that produces the optimal objective function value for all objectives simultaneously. In this case the noninferior set is a single criterion vector, and the efficient set is a single solution (unless there are alternative optima, in which case they are all efficient solutions). This criterion vector is referred to as the *ideal criterion vector.* In most multiobjective problems the ideal criterion vector is infeasible, but it is often used as a reference point when

comparing efficient solutions.

The theory for multiobjective (MOP), or vector optimisation (VOP), problems embraces both the linear and nonlinear cases. This discussion will first consider linear problems (MOLP or LVOP), and then consider the effects of including integer variables (the MOIP). The following results are well established and will be given without proofs. A detailed discussion, with proofs, can be found in Chankong & Haimes (1983), (see also Cohon 1978, Steuer 1986).

For linear problems with linearly independent objective functions, all of the efficient solutions lie on the frontier of the set of feasible solutions, X. Further, the set of efficient solutions, X^*, consists of faces and edges that are characterised by the efficient extreme points, and all points in X^* are connected. However, because decision makers frequently have nonconcave utility functions, the X^* can be nonconvex, and the optimal point can occur at a nonextreme point. Nonconvex utility functions also mean that an MOLP can have local optima that are not global optima. In addition, there may be a finite number of alternative optima that form a disconnected set in solution space.

When the problem includes integer variables (and becomes an MOIP) the noninferior set need not be connected. As with single objective optimisation, MOIP's are much harder to solve than MOLP's. However, unlike the branch and bound algorithm used to solve continuous integer programs, no general technique has been developed to handle integer problems with multiple objectives.

Techniques for solving multiobjective optimisation problems can be grouped into two broad categories : *generating techniques* and *methods that incorporate preferences*, although there is some overlap between them.

Set generation techniques find the set of noninferior criterion vectors, either exactly or approximately, and the decision maker chooses a criterion vector, and thus a solution, from this set. This approach has the advantage that the decision maker does not need to articulate his/her preferences explicitly, although they will be expressed implicitly by the final choice of solution. The main disadvantage is that set generation techniques can proceed without much input from the decision maker, and this lack of involvement may make the decision maker reluctant to implement the results. The other disadvantage is that a large computational effort is required to generate the full noninferior set, much of which will be of no interest to the decision maker. However, it may be possible for decision makers to provide

a partial statement of their preferences that can be used to reduce the size of the feasible region, and thus the size of the noninferior set and the computational effort required to generate it.

Interactive techniques (or methods that incorporate preferences) start by finding one criterion vector, or a small number of criterion vectors, and presenting them to the decision maker. The decision maker examines these results and provides feedback to the algorithm. When an algorithm provides a single criterion vector the feedback may be a statement about which of the objectives should be relaxed or tightened. When several criterion vectors are provided, the decision maker may make a statement about which criterion vectors are preferred to the others. The algorithm uses the feedback as a basis for searching for another criterion vector, or vectors, for the decision maker to consider. The process continues until the decision maker is willing to accept a criterion vector, and thus a solution, as being satisfactory.

Set generation and interactive techniques are outlined in Sections 3.3 and 3.4

3.3 Set Generation Techniques

This section will review the two classical set generation techniques that form the basis of the approach to scenario analysis that is proposed in this thesis.

3.3.1 Weighting Method

Problem (3.1), has a corresponding weighting problem $P(\mathbf{w})$:

$$\text{maximise} \qquad \mathbf{w}C\mathbf{x} \tag{3.2a}$$

$$\text{subject to} \qquad \mathbf{x} \in X \tag{3.2b}$$

where: $C = K \times n$ objective function matrix

$\mathbf{w} = K \times 1$ vector, such that: $\mathbf{w} \geq \mathbf{0}$, $\sum_{k=1}^{K} w_k = 1$

In the weighting method the weighting vector, $\mathbf{w}$, is varied parametrically to find the noninferior extreme points that characterise the noninferior set, N. Although this is not an efficient method for finding an exact representation of the efficient

set, it is used to find an approximation to the set. Generally the analysis starts by optimising the objectives individually by solving problem $P(\mathbf{w})$ using a weighting vector of the form: $(0, 0, \ldots, 1, \ldots, 0, 0)$. This optimises for one objective. The value for this objective is then fixed at its optimal value and problem $P(\mathbf{w})$ is solved again using the weighting vector $(1, 1, \ldots, 1, \ldots, 1, 1)$. The fixed objective value is then released and the steps repeated for the next objective. This process finds the "end points" of the noninferior set, N. That is, an optimal solution, and its corresponding criterion vector, is found for each objective.

Intermediate points are found by solving problem (3.2) for a series of weighting vectors. Each weighting vector will find an extreme point in solution space, and in criterion space. The extreme points in criterion space can be connected together to form an approximation of the noninferior set. An initial analysis may be carried out using a small number of weighting vectors. The decision maker can then decide which regions of the approximation are of particular interest, and these regions can be refined using new weighting vectors. When used in this fashion, the weighting method becomes an approach that incorporates preferences.

Cohon (1978) and Cohon et al. (1979) develop an implementation of the weighting method for bicriterion problems, that they call the *the noninferior set estimation (or NISE) method.* The method finds a lower bound approximation to the noninferior set using an iterative approach. At each iteration, an upper and lower bound is found for the noninferior set, and the maximum possible distance between the bounds is calculated for each line segment in the lower bound. This distance is referred to as the *maximum possible error* or MPE. The line segment with the greatest MPE is used to generate the next noninferior point. The method continues until the largest MPE is within a predetermined tolerance, or an iteration limit is reached.

Solanki et al. (1993) extend the NISE method to higher dimensions (and call the method XNISE).

3.3.2 Constraint Method

The constraint method finds an approximation to the noninferior set by appending constraints to problem (3.2) to impose lower bounds on all but one of the objectives. The problem is then optimised for the remaining objective. Thus problem (3.2)

becomes:

$$\begin{aligned}
&\text{maximise} && z_i && && (3.3\,\text{a})\\
&\text{subject to} && \mathbf{x} \in X && && (3.3\,\text{b})\\
& && z_k = \mathbf{c}_k\mathbf{x} && \text{for } k = 1, 2, \ldots, K && (3.3\,\text{c})\\
& && z_k \geq z_k^{lb} && \text{for } k = 1, 2, \ldots, K; \quad k \neq i && (3.3\,\text{d})
\end{aligned}$$

where: z_i = objective to be maximised

z_k^{lb} = value which must be attained by objective k

As with the weighting method, the analysis generally starts by finding the "end points" of the noninferior set, N. This is conveniently done in the same way as is used for the weighting method. Intermediate points are found by using a grid of z_k^{lb} values. The constraint method cannot determine the noninferior set exactly, because the intersection of constraints (3.3 d) with constraints (3.3 b) will not usually occur at extreme points. For problems with more than two objectives, some of the instances of problem (3.3) will be infeasible. However, it does permit the decision maker to concentrate on regions of particular interest by varying the width of the intervals in the grid.

In a similar fashion to the weighting method, a coarse grid of constraints can be used to generate an initial approximation, and then a finer grid used to refine areas of interest.

3.4 Interactive Techniques

Although the work presented in this thesis is based on set generation techniques, this section will round out the review of multiobjective optimisation by considering a technique that incorporates preferences. As outlined at the end of Section 3.2, techniques that incorporate preferences start by finding a sample set of noninferior criterion vectors. These vectors are presented to the decision maker, who provides feedback to the algorithm. The feedback is used to influence the choice of the next sample, and the process continues until the decision maker decides that one of the solutions presented to him/her is satisfactory. The interactive technique presented here is the Tchebycheff Procedure of Steuer & Choo (1983).

The Tchebycheff procedure starts by finding the *ideal criterion vector*, $\mathbf{z}^{**} \in \Re^K$, such that $z_k^{**} = max\{\mathbf{c}_k\mathbf{x}|\mathbf{x} \in X\} + \epsilon_k$, where $\epsilon_k \geq 0$. $\epsilon_k = 0$ is permissible unless z_k^{**} appears in more than one noninferior criterion vector, or the only noninferior criterion vector that includes z_k^{**} also includes one, or more, of z_j^{**}, $j \neq k$. When $\epsilon_k > 0$, it is set to a small positive scalar value.

The ideal criterion vector is an infeasible point in criterion space that can be "aspired to". That is, solutions with criterion vectors that are close (in some sense) to the ideal criterion vector are preferred over solutions with criterion vectors that are further away.

The distance between any criterion vector, $\mathbf{z}$, and the ideal criterion vector, $\mathbf{z}^{**}$ is measured using the *augmented weighted Tchebycheff metric*:

$$|||\mathbf{z}^{**} - \mathbf{z}|||_x^\lambda = ||\mathbf{z}^{**} - \mathbf{z}||_x^\lambda + \rho\, \epsilon^T(\mathbf{z}^{**} - \mathbf{z})$$

$$\text{where: } ||\mathbf{z}^{**} - \mathbf{z}||_x^\lambda = \max_{i=1,\ldots,K}\{|\mathbf{z}^{**} - \mathbf{z}|\}$$

$$\text{and } \lambda \in \Lambda = \{\lambda \in \Re^K \mid \lambda_i \geq 0,\ \textstyle\sum_{i=1}^K \lambda_i = 1\}$$

After the ideal criterion vector has been found, a widely dispersed group of λ weighting vectors is formed. For each λ, a criterion vector is found that minimises the distance from the ideal criterion vector, according to the augmented, weighted Tchebycheff metric. These criterion vectors are presented to the decision maker, who is asked to identify the most preferred one. The chosen criterion vector is used to determine a new set of weighting vectors that are more concentrated around the chosen criterion vector, than the first set was. This set of weighting vectors is used to find a new set of criterion vectors for the decision maker to choose among. The chosen criterion vector is used to determine another set of weighting vectors that are more concentrated than before. Thus, at each iteration, the algorithm concentrates on a progressively smaller region of the noninferior set. The procedure continues until the decision maker decides that one of the criterion vectors is an acceptable solution to the problem.

The use of the augmented, weighted Tchebycheff metric ensures that only non-dominated criterion vectors will be presented to the decision maker. An important characteristic of this approach is that it can identify solutions that are not extreme

points. This is important when the available alternatives are presented to the decision maker as a sample of discrete points, as is done by interactive methods, otherwise it would be impossible for the decision maker to find and choose a non-extreme criterion vector. It should be noted that, although set generation techniques identify extreme points only, these points are used to describe the surface of the noninferior set and non-extreme points can be identified by interpolating between them.

3.5 Integer Multiobjective Problems

As with single objective optimisation, the inclusion of integer variables makes multiobjective optimisation problems much harder to solve. In fact, the increase in complexity and computational effort required to solve multiobjective problems when integer variables are added is much greater than the increase observed for single objective problems. This is largely due to the need, with multiobjective integer problems, to compare vectors of objective values when determining whether a node in the branch and bound tree can be fathomed. Not only is it much harder to fathom nodes in the tree when working with multiple objectives, but, for a full enumeration of the noninferior set the branch and bound tree has to be evaluated to the bottom of every branch that produces a nondominated solution. This increases the storage requirements, as does the fact that the incumbent is no longer a single objective function value, but a list of the criterion vectors that have not yet been proved to be inferior. Because no single criterion vector will dominate all of the problem's inferior criterion vectors, all criterion vectors that have not yet been shown to be inferior must be retained in the incumbent list to make the test for inferiority at a node as discriminating as possible.

It is, therefore, not surprising that the development of solution methods for the multiobjective integer linear problem (MOILP) has been slow and often directed at problems with special structures that can be exploited by the solution process. Early work concentrated on pure integer problems with binary variables only, for example, Pasternak & Passy (1973) present a procedure for bicriterion problems with binary variables. Bitran (1977*a*) and (1977*b*) proposes a method for determining the efficient solutions for problems in which all variables are binary, and in which there are more than two objectives. Kiziltan & Yucaoğlu (1983) present a method based on implicit enumeration techniques, and develop domination tests designed to improve

fathoming of the branch and bound tree.

Other authors (e.g. Klein & Hannan 1982) have developed algorithms to find the whole efficient set for general integer problems, that is, problems in which all variables are integer, but need not be binary.

As with continuous MOLP's, interactive methods have been developed in which a few solutions are generated and the decision maker is asked to choose between them. This choice is used to guide the search for another set of solutions to choose among, and the process terminates when the decision maker is either satisfied, or he/she decides that enough effort has been expended. For example, Marcotte & Soland (1986) present an interactive algorithm based on a branch and bound algorithm, while Karaivanova, Narula & Vassilev (1993) propose a modification of Steuer's Interactive Tchebycheff Procedure (described in Steuer 1986).

A striking result of searching the literature is the small number of papers that report work done on mixed integer multiobjective problems. Almost all of the work found pertains to pure integer linear problems only. This conclusion is supported up to 1986 by Teghem & Kunsch (1986) who report that "In our knowledge, no specific methods have been published to characterise the set of efficient solutions for multi-objective mixed integer linear programming". However, Ramesh, Zionts & Karwan (1986) propose an interactive approach that combines a branch and bound algorithm with the determination of the decision maker's preferences.

This approach starts by relaxing the integrality constraints and using the Zionts and Wallenius procedure (Zionts & Wallenius 1983) to determine a weighting vector that is consistent with the decision maker's preferences, and a solution that is an optimal solution for that weighting vector. If this solution is integer, the problem is solved. If it is not integer the procedure uses a branch and bound algorithm to find the preferred integer solution. At each node the algorithm either uses the Zionts and Wallenius procedure to solve the subproblem and find a new weighting vector, or it branches to form a new subproblem that is solved using the current weighting vector. If the subproblem is solved at every node, the authors refer to it as the "BTI" algorithm (Branch Till Integrality). This strategy only presents integer solutions to the decision maker, and so minimises the number of questions asked of the decision maker, but at a high computational cost. The strategy of always solving the subproblem at a node is referred to as the ZWA algorithm (perform the Zionts and Wallenius procedure for All candidate problems) and it questions

the decision maker about non-integer solutions, thus requiring more input from the decision maker, but reducing the computational cost. The authors suggest that the BTI strategy is preferable to avoid excessive questioning of the decision maker, but if the computational effort is too large, a hybrid approach is recommended in which some nodes are branched and some are solved.

Mavrotas, Diakoulaki & Papayannakis (1999) report a branch and bound algorithm for solving mixed integer multiobjective optimisation problems. In this approach a branch and bound tree is searched to find all of the noninferior solutions. At each node an ideal criterion vector is found by optimising for each objective separately. If this ideal criterion vector is dominated by the criterion vector of a known solution that satisfies the integrality requirements, then the node can be fathomed. Thus the single incumbent value used in the single objective MIP is replaced by a list of criterion vectors, some, or all, of which may turn out to be noninferior. The authors report the application of their algorithm to a small problem taken from the Greek energy sector. Although the algorithm reported in this paper is similar to the algorithm developed in this thesis, it came to our notice after we had completed our work. We also note that the example reported by Mavrotas et al. (1999) has only two objectives, whereas we have applied our algorithm to a problem with five objectives.

Finally, it should be noted that the augmented weighted Tchebycheff metric discussed in Section 3.4 can be applied to discrete problems. However, it will also exact a high computational burden, because several integer problems have to be solved at every iteration.

3.6 Summary

In this chapter we briefly described the multiobjective optimisation problem, and two set generation techniques that are used to solve them, the weighting method and the constraint method. Noninferior Set Scenario Analysis (NSSA) proposed in this thesis is based on these set generation techniques. We have rounded out the discussion by outlining the Tchebycheff Procedure as an example of a technique that incorporates preferences (an interactive method). We have chosen not to use an interactive method for this work, because these methods concentrate on finding a solution to implement, whereas our work is designed to provide decision makers

with a "picture" of the available trade-offs.

Because this thesis also develops an approach for dealing with decision making problems that include binary stage one variables, we have considered the addition of integer variables to the multiobjective optimisation problem, and the resulting increase in the computational effort required to find a solution.

Chapter 4

The Theoretical Basis of Noninferior Set Scenario Analysis

4.1 Introduction

In this chapter we discuss the theoretical basis of Noninferior Set Scenario Analysis. We start by describing the problem situation in which the decision maker is faced with an uncertain future, but must implement a decision now, before the future uncertainties are resolved. We model the problem situation as a two-stage problem in which the future uncertainty is modelled as a set of scenarios, only one of which will occur. After the outcome of the uncertainty has been observed, the decision maker will have an opportunity to adjust the situation by implementing a recourse decision.

Having described the problem we review how it can be formulated as a scenario analysis problem, and as a stochastic optimisation problem. We then develop our approach in which each scenario is viewed as having its own objective. These objectives are in conflict because a stage one decision that prepares well for one scenario will prepare poorly for at least one of the others. This view of the problem leads us to formulate it as a multiobjective optimisation problem in which good performance under some scenarios can only be obtained by accepting a poor performance under other scenarios.

We apply the set generation technique of Cohon (1978) and Cohon et al. (1979) to this problem to find an approximation to the noninferior set, and thus a set of alternative stage one decisions for the decision maker to choose among. This

formulation means that the problem can be analysed without assuming probabilities for the scenarios. More importantly, it provides a systematic method for identifying a set of alternative decisions for the decision maker to choose among, thus overcoming the short-coming of stochastic optimisation that only provides the decision maker with a single solution.

We then show how this approximation to the noninferior set can be interpreted to provide the solution for the stochastic optimisation formulation for any choice of scenario probabilities. This approximation also provides sensitivity analysis of the scenario probabilities for the stochastic optimisation formulation.

In Section 4.2 we describe the problem situation, and follow, in Section 4.3, with the scenario analysis and stochastic optimisation formulations of the problem. In Section 4.4 the NSSA formulation is developed and related to the stochastic optimisation formulation. In Section 4.5 we set out definitions and notation for later use, and in Section 4.6 we discuss some assumptions that NSSA makes about the decision maker. In Section 4.7 NSSA is developed for problems with two scenarios, and the interpretation of the results of the analysis is presented. The chapter is summarised in Section 4.8.

4.2 Description of the Problem

In this section we set the scene by formulating the problem faced by strategic decision makers as a two stage problem in which a decision is to be implemented now, followed by a lead time during which the organisation must continue to operate. At a future point in time conditions are expected to change in an uncertain manner. This change may involve a change in the organisation's objectives, changes in its operating environment, or both. Once the uncertainty has been resolved, the decision maker will have an opportunity to make adjustments before operating in the new environment. These stage two (or recourse) decisions enable the decision maker to respond to the resolution of the uncertainty. It is, in some way, preferable (or in fact, necessary), for the decision maker to implement a decision now, and then make adjustments after the resolution of the uncertainty, than it is to wait and see what happens before acting. If this was not the case, the decision maker would wait until the uncertainty had been resolved, and then decide what to do. We will refer to the decisions to be taken during the first stage as the *stage one* decisions, and the

decisions to be taken after the resolution of uncertainty as the *recourse* decisions. The decision maker may also take operating decisions during stage one, and after the resolution of the uncertainty.

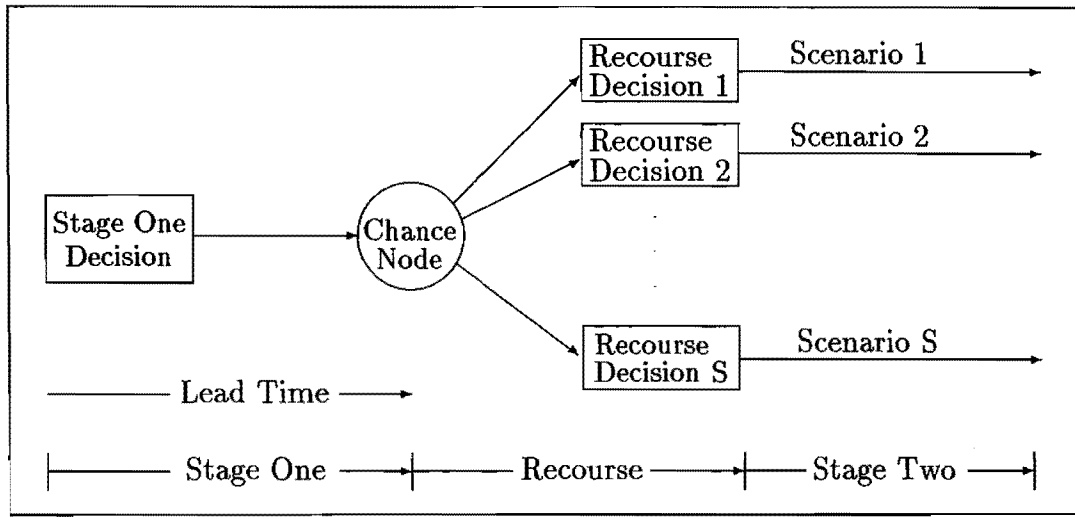

Figure 4.1: Structure of the Problem
Because of its lead time the stage one decision must be taken now, before the outcome of the chance node can be observed. The organisation must continue to operate during stage one. Recourse decisions can be taken once the uncertainty has been resolved.

In scenario planning the uncertainty is modelled as a small number of scenarios, each of which is a description of how the uncertainty may turn out. These scenarios do not provide a description of all possibilities, rather they are designed to be representative examples of what may happen. The decision structure is modelled as a stage one decision to be taken now, followed by a set of recourse decisions, one for each scenario, which are to be taken once the decision maker knows which scenario is occurring. We will denote the set of scenarios as Ω, with S members ω_s, that is $\Omega = \cup_{s=1}^{S} \omega_s$. The structure of the problem is shown in Figure 4.1.

The problem of Figure 4.1 consists of two decision making problems. In the stage one problem, the decision maker must decide what to do now, and in the recourse problem, the decision maker must take a decision in light of the resolution of the uncertainty. The recourse problem is a function of the stage one decision, the outcome of the uncertainty, and of the stage one operating decisions. The stage one

problem can be represented as:

$$\begin{aligned} &\text{maximise} && z^I(\mathbf{x}) = \mathbf{c}_0\mathbf{x} \\ &\text{subject to} && A\mathbf{x} = \mathbf{b}_0 \\ & && \mathbf{x} \geq 0 \end{aligned} \tag{4.1}$$

where: z^I = the objective function value for stage one only

$\mathbf{x}$ = the vector of stage one decisions

$\mathbf{c}_0$ = the vector of stage one costs and benefits of the stage one decisions

A = the matrix of stage one constraint coefficients

$\mathbf{b}_0$ = the right hand side vector for stage one

Once the uncertainty has been resolved, and the decision maker knows which scenario is occurring, the recourse problem will be one of the S problems:

$$\begin{aligned} &\text{maximise} && z^R_\omega(\mathbf{x}) = \mathbf{c}_\omega\mathbf{x} + \mathbf{q}_\omega\mathbf{y}_\omega \\ &\text{subject to} && B_\omega\mathbf{y}_\omega = \mathbf{b}_\omega - T_\omega\mathbf{x} \\ & && \mathbf{y}_\omega \geq 0 \end{aligned} \tag{4.2}$$

where: ω = the scenario which has occurred, $\omega \in \Omega$

$z^R_\omega(\mathbf{x})$ = recourse objective function value, when the stage one decision is $\mathbf{x}$

$\mathbf{c}_\omega$ = vector of stage two costs and benefits of the stage one decision vector

$\mathbf{y}_\omega$ = vector of recourse decisions and stage two operating decisions

$\mathbf{q}_\omega$ = vector of costs and benefits of the stage two decisions

B_ω = matrix of constraint coefficients for the stage two decisions

$\mathbf{b}_\omega$ = right hand side vector for stage two

T_ω = matrix of interactions of $\mathbf{x}$ with the stage two decisions

Note: $\mathbf{c}_\omega\mathbf{x}$ is constant in the recourse problem, but is included to enable different stage one decisions to be compared on the basis of their performance under the scenarios.

The decision maker could ignore the future, and solve problem (4.1), with the intention of dealing with problem (4.2) when it eventuates. Normally, however, a decision maker will want to adjust the stage one decision to provide for the future by including the interactions between problems (4.1) and (4.2) in the analysis. The decision maker must balance the need to obtain a good result at stage one against the need to perform well at stage two. The decision maker will also face competing demands for good performance under each scenario. Generally, there will be many feasible stage one decisions that will perform well under some scenarios, and poorly under others, but no stage one decisions that perform well under all scenarios. The decision maker must find a stage one decision that leads to acceptable trade-offs between all of the scenarios. That is, the decision maker trades off poor performance under some scenarios against better performance under other scenarios. It is likely that there is no feasible solution that produces trade-offs that are entirely satisfactory to the decision maker, in which case he or she will have to accept the best available compromise.

It is important to recognise that the decision maker is trading off possibilities, not certainties. In the event, the decision maker will get the performance of whichever scenario actually occurs, and not the performance that would have been observed under the other scenarios. This means that when the decision maker chooses a solution that performs better under one scenario than another, he or she is trading off the *possibility* of getting the better outcome, against the *possibility* of getting the poorer outcome. He or she will not get both.

4.3 Scenario Analysis and Stochastic Optimisation

A commonly used approach is to optimise the stage one decision vector for each scenario in turn. The decision maker formulates a deterministic optimisation problem for each scenario, that is S versions of the following problem:

$$\text{maximise} \qquad z_\omega = (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x}_\omega + \mathbf{q}_\omega\mathbf{y}_\omega \tag{4.3 a}$$

$$\text{subject to} \qquad A\mathbf{x}_\omega = \mathbf{b}_0 \tag{4.3 b}$$

$$B_\omega\mathbf{y}_\omega + T_\omega\mathbf{x}_\omega = \mathbf{b}_\omega \tag{4.3 c}$$

$$\mathbf{y}_\omega \geq 0,\ \mathbf{x}_\omega \geq 0 \tag{4.3 d}$$

where: ω = the scenario being considered, $\omega \in \Omega$

z_ω = the objective function value for scenario ω

$\mathbf{x}_\omega$ = vector of stage one decisions optimised for scenario ω

$\mathbf{c}_0$ = vector of stage one costs and benefits of the stage one decisions

$\mathbf{c}_\omega$ = vector of stage two costs and benefits of the stage one decisions

A = matrix of stage one constraint coefficients

$\mathbf{b}_0$ = right hand side vector for stage one

$\mathbf{y}_\omega$ = vector of recourse decisions and stage two operating decisions

$\mathbf{q}_\omega$ = vector of costs and benefits of the stage two decisions

B_ω = matrix of constraint coefficients for the stage two decisions

$\mathbf{b}_\omega$ = right hand side vector for stage two

T_ω = matrix of the interactions of $\mathbf{x}$ with the stage two decisions

These problems will produce S "optimal" stage one decisions, $\mathbf{x}^*_\omega$, one for each scenario. These scenario-optimal decisions will usually be different for each scenario, and some of them may be infeasible under another scenario. Feasibility under all scenarios may be ensured by including the constraints for all scenarios when optimising for each scenario.

That is, constraint (4.3 c) is replaced by: $B_k\mathbf{y}_k + T_k\mathbf{x}_\omega = \mathbf{b}_k \quad \forall \quad k \in \Omega$, which

converts problem (4.3) into problem (4.4).

$$\begin{aligned}
\text{maximise} \quad & z_\omega = (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x}_\omega + \mathbf{q}_\omega\mathbf{y}_\omega && && (4.4\,\text{a})\\
\text{subject to} \quad & A\mathbf{x}_\omega = \mathbf{b}_0 && && (4.4\,\text{b})\\
& B_k\mathbf{y}_k + T_k\mathbf{x}_\omega = \mathbf{b}_k && \forall\ k \in \Omega && (4.4\,\text{c})\\
& \mathbf{x}_\omega \geq 0 && && (4.4\,\text{d})\\
& \mathbf{y}_k \geq 0 && \forall\ k \in \Omega && (4.4\,\text{e})
\end{aligned}$$

If there is no stage one solution that is feasible under all scenarios, then problem (4.4) will be infeasible. However, even if problem (4.4) is feasible, the decision maker still does not have a stage one decision to implement. Using the terminology of Rockafellar & Wets (1991), the stage one decision vectors, $\mathbf{x}_\omega$, are not implementable, because the decision maker does not yet know which scenario, ω, will occur. Her choice of a decision to implement cannot be conditioned on knowing ahead of time, which scenario is going to occur. Nevertheless, she could implement one of $\mathbf{x}^*_\omega$, or she could look for a compromise stage one decision that will achieve an acceptable balance between the scenarios, even though it will not prepare optimally for any of them.

If the decision maker can specify her utility function for the problem, then this utility function can be used as the objective function for problem (4.4), and the problem solved to find her preferred decision. This is the approach taken when the problem is formulated as a stochastic optimisation problem. The decision maker is assumed to be risk neutral with a utility that is maximised when the expected value over the scenarios is maximised. Thus a single objective function is formed from the weighted sum of the scenario objectives, where the scenario probabilities are used as the weights. The problem becomes:

$$\begin{aligned}
\text{maximise} \quad & z = \Big(\mathbf{c}_0 + \sum_{\omega\in\Omega} p_\omega \mathbf{c}_\omega\Big)\mathbf{x} + \sum_{\omega\in\Omega} p_\omega \mathbf{q}_\omega \mathbf{y}_\omega && && (4.5\,\text{a})\\
\text{subject to} \quad & A\mathbf{x} = \mathbf{b}_0 && && \\
& B_\omega\mathbf{y}_\omega + T_\omega\mathbf{x} = \mathbf{b}_\omega && \forall\ \omega \in \Omega && (4.5\,\text{b})\\
& \mathbf{x} \geq 0 && && \\
& \mathbf{y}_\omega \geq 0 && \forall\ \omega \in \Omega &&
\end{aligned}$$

where: p_ω = the probability that scenario ω will occur

Formulation (4.5) is the familiar two-stage stochastic optimisation problem. The stage one decision is chosen to maximise the expected value of the objectives over all of the scenarios. However, if there are large variations in the performance of this stage one decision under the different scenarios, then the expected value optimum may be unacceptable to a risk averse decision maker. Similarly, a risk taking decision maker may prefer to implement a decision tailored for success under a particular scenario, and accept the poor outcome that will result if events turn out otherwise.

If the decision maker cannot specify her utility function for the problem, then she needs a method that will find nondominated, feasible alternatives to choose among. In this context, a stage one decision is nondominated if no other feasible decision exists that performs better under at least one scenario, and no worse under the others. The stage one decision found by problem (4.5) is nondominated and implementable. However, it is only one decision, and the decision maker is given little information on alternatives that may be available.

As discussed in Section 2.7, the stochastic optimisation problem has been reformulated to find an optimal solution that is less sensitive to the outcome of the uncertainty. Mulvey et al. (1995) and Dembo (1991) append penalty terms to the objective, so that the performance of the resulting "robust" optimal solution varies less between the scenarios than does the optimal solution of the standard formulation. Eppen et al. (1989) append risk limitation constraints to the problem to control the downside risk. However, these approaches still produce a single decision that the decision maker must accept or reject.

In this work we propose a different approach in which the problem is formulated as a multiobjective optimisation problem, and a set of nondominated solutions is found. The decision maker can then choose among these solutions according to her risk preferences.

4.4 Noninferior Set Scenario Analysis

In this approach we choose to view each scenario as having a separate objective, and to bring them together as a Multiobjective Optimisation problem. We gather the scenario analysis problems (4.4) into the multiobjective problem (4.6), and find

an approximation to the noninferior set in objective space. The aim is to provide the decision maker with a description of the available alternatives in terms of their performance under all of the scenarios. The decision maker can then look for an alternative that provides an acceptable trade-off between the scenarios.

$$\text{maximise} \quad \left\{ \begin{array}{ll} z_1 = & (\mathbf{c}_0 + \mathbf{c}_1)\mathbf{x} + \mathbf{q}_1\mathbf{y}_1 \\ z_2 = & (\mathbf{c}_0 + \mathbf{c}_2)\mathbf{x} + \mathbf{q}_2\mathbf{y}_2 \\ \vdots & \vdots \\ z_S = & (\mathbf{c}_0 + \mathbf{c}_S)\mathbf{x} + \mathbf{q}_S\mathbf{y}_S \end{array} \right\} \tag{4.6 a}$$

$$\begin{aligned} \text{subject to} \quad A\mathbf{x} &= \mathbf{b}_0 \qquad\qquad (4.6\,\text{b}) \\ B_1\mathbf{y}_1 + T_1\mathbf{x} &= \mathbf{b}_1 \\ B_2\mathbf{y}_2 + T_2\mathbf{x} &= \mathbf{b}_2 \\ \vdots \qquad\quad & \vdots \\ B_S\mathbf{y}_S + T_S\mathbf{x} &= \mathbf{b}_S \\ \mathbf{x} &\geq 0 \\ \mathbf{y}_\omega &\geq 0 \qquad \forall\ \omega \in \Omega \end{aligned}$$

Because the decision maker needs an implementable stage one decision, the *nonanticipativity restriction* (this term was coined by Rockafellar & Wets (1991), see Section 2.7.3) is imposed in problem (4.6) by using a single stage one decision vector, $\mathbf{x}$, in place of the scenario specific stage one decision vectors, $\mathbf{x}_\omega$, of problem (4.4). The stage one decision must also be feasible under all scenarios, so the scenario specific constraints are gathered together into a single constraint set.

This formulation can be expressed as the weighting problem:

$$\text{maximise} \quad z = w_1 z_1 + w_2 z_2 + \ldots + w_S z_S \tag{4.7 a}$$

$$\begin{aligned} \text{subject to} \quad z_\omega &= (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x} + \mathbf{q}_\omega\mathbf{y}_\omega \qquad \forall\ \omega \in \Omega \\ A\mathbf{x} &= \mathbf{b}_0 \qquad\qquad (4.7\,\text{b}) \\ B_\omega\mathbf{y}_\omega + T_\omega\mathbf{x} &= \mathbf{b}_\omega \qquad \forall\ \omega \in \Omega \\ \mathbf{x} &\geq 0 \\ \mathbf{y}_\omega &\geq 0 \end{aligned}$$

where: w_ω = the weight placed on the objective of scenario ω

This is, of course, the two-stage stochastic optimisation problem (4.5). However, it is presented here as a weighting of the separate scenario objective functions, to emphasise the change from viewing the problem as having a single objective, to that of having multiple, competing objectives. This formulation of the objective function is also necessary because alternative solutions will be compared according to their criterion vectors (vectors of scenario objective function values), and not according to a single summary objective function value.

4.5 Definitions and Notation

Before developing the theoretical basis of Noninferior Set Scenario Analysis, we will define some concepts and associated notation.

Definition 4.1 *A stage one decision vector* $\mathbf{x}$ *is said to be* scenario-feasible *if it is feasible under at least one of the scenarios* $\omega \in \Omega$. *That is, there exists a recourse vector,* $\mathbf{y}_\omega$, *such that* $(\mathbf{x}, \mathbf{y}_\omega)$ *is a feasible solution to problem* (4.3) *for scenario* ω.

A scenario-feasible stage one decision vector may be infeasible under one, or more, of the other scenarios, in which case it is an infeasible solution to problem (4.4).

Definition 4.2 *A scenario-feasible stage one decision vector* $\mathbf{x}$ *is said to be* scenario-optimal *if there exists a recourse vector,* $\mathbf{y}_\omega$, *such that* $(\mathbf{x}, \mathbf{y}_\omega)$ *is an optimal solution to problem* (4.3) *for scenario* ω. *A scenario-optimal stage one decision vector is denoted as* $\mathbf{x}^*_\omega$, *and the corresponding solution to problem* (4.3) *is denoted as* $(\mathbf{x}^*_\omega, \mathbf{y}^*_\omega)$. *The optimal objective function value for scenario* ω *is denoted as* z^*_ω, *where* $z^*_\omega = (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x}^*_\omega + \mathbf{q}_\omega\mathbf{y}^*_\omega$. *Solution* $\mathbf{x}^*_\omega$ *need not be unique, and there may be more than one optimal recourse vector for any particular* $\mathbf{x}^*_\omega$. *However,* z^*_ω *is unique.*

Definition 4.3 *A scenario-feasible, stage one decision vector* $\mathbf{x}$ *is* problem-feasible *if it is feasible under all scenarios. That is, for every scenario* $\omega \in \Omega$, *there exists a recourse vector,* $\mathbf{y}_\omega$, *such that* $(\mathbf{x}, \mathbf{y}_\omega)$ *is a feasible solution to problem* (4.3) *for that scenario.*

A problem-feasible stage one decision vector is a feasible solution to problem (4.4).

Definition 4.4 *A problem-feasible stage one decision vector* $\mathbf{x}$ *is said to be* scenario-maximal *if it is an optimal stage one decision for at least one scenario* ω. *That is, for every scenario* $j \in \Omega$, *there exists a recourse vector,* $\mathbf{y}_j$, *such that* $(\mathbf{x}, \mathbf{y}_j)$ *is a feasible solution to problem* (4.3) *for that scenario, and the solution* $(\mathbf{x}, \mathbf{y}_\omega)$ *is the optimal solution for scenario* ω. *A scenario-maximal stage one decision vector is denoted as* $\mathbf{x}_\omega^{max}$, *and the corresponding solution to problem* (4.3) *is denoted as* $(\mathbf{x}^{max}, \mathbf{y}_\omega^{max})$. *The optimal objective function value for scenario* ω *is denoted as* z_ω^{max}, *where* $z_\omega^{max} = (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x}_\omega^{max} + \mathbf{q}_\omega \mathbf{y}_\omega^{max}$. *Vector* $\mathbf{x}_\omega^{max}$ *need not be unique, and there may be more than one optimal recourse vector for any particular* $\mathbf{x}_\omega^{max}$. *However,* z_ω^{max} *is unique.*

Definition 4.5 *A problem-feasible stage one decision vector* $\mathbf{x}$ *is an* efficient *stage one decision vector if there does not exist another problem-feasible stage one decision vector* $\mathbf{u}$ *such that*

$$z_\omega(\mathbf{u}) \geq z_\omega(\mathbf{x}) \qquad \text{for all } \omega \in \Omega$$

and

$$z_\omega(\mathbf{u}) > z_\omega(\mathbf{x}) \qquad \text{for at least one } \omega \in \Omega$$

For convenience of notation, the vector relationship

$$\mathbf{V_1} \geq \mathbf{V_2}$$

is used to mean that the inequality holds when each element of one vector is compared with its corresponding element in the other vector. Clearly the dimensions of the vectors must be the same. Thus the expression $\mathbf{Z}(\mathbf{u}) \geq \mathbf{Z}(\mathbf{x})$ is equivalent to:

$$z_\omega(\mathbf{u}) \geq z_\omega(\mathbf{x}) \qquad \text{for all } \omega \in \Omega$$

Let X^S denote the set of scenario-feasible stage one decision vectors i.e.

$$X^S = \{\mathbf{x} \mid \mathbf{x} \text{ is feasible in problem (4.3) for at least one of } \omega \in \Omega\}$$

Let X^P denote the set of problem-feasible stage one decision vectors i.e.

$$X^P = \{\mathbf{x} \mid \mathbf{x} \text{ is feasible in problem (4.3) for all of } \omega \in \Omega\}$$

or equivalently:

$X^P = \{\mathbf{x} \mid \mathbf{x} \text{ is feasible in problem (4.4)}\}$

Let X^E denote the set of problem-feasible stage one decision vectors that are associated with the extreme points in decision space of problem (4.4)

Let X^M denote the set of scenario-maximal stage one decision vectors

Let X^N denote the set of efficient stage one decision vectors

Let $\mathbf{Z}^*$ denote the vector of scenario-optimal objective values i.e.

$\mathbf{Z}^* = (z_1^*, z_2^*, \ldots, z_S^*)$

normally $\mathbf{Z}^*$ is infeasible.

Let $\mathbf{Z}^{max}$ denote the vector of scenario-maximal objective values i.e.

$\mathbf{Z}^{max} = (z_1^{max}, z_2^{max}, \ldots, z_S^{max})$

$\mathbf{Z}^{max}$ is referred to as the *ideal* objective vector. Normally it is infeasible.

Let $\mathbf{Z}^{min}$ denote the vector of minimum objective values over the noninferior set i.e.

$\mathbf{Z}^{min} = (z_1^{min}, z_2^{min}, \ldots, z_S^{min})$

Where $z_\omega^{min} = \underset{\mathbf{x} \in X^N}{\text{minimum}}\, z_\omega(\mathbf{x})$ for all $\omega \in \Omega$

$\mathbf{Z}^{min}$ is feasible, but inferior, unless $\mathbf{Z}^{max}$ is feasible, in which case X^N is a singleton (unless there are alternative optima),

and $\mathbf{Z}^{min} = \mathbf{Z}^{max}$

Let Q^P denote the set of problem feasible objective vectors i.e.

$Q^P = \{\mathbf{Z}(\mathbf{x}) \mid \mathbf{x} \in X^P\}$

Let Q^E denote the set of extreme points in objective space

$Q^E = \{\mathbf{Z}(\mathbf{x}) \mid \mathbf{x} \in X^E\}$

Let Q^M denote the set of scenario-maximal objective vectors i.e.

$Q^M = \{\mathbf{Z}(\mathbf{x}) \mid \mathbf{x} \in X^M\}$

Let $\mathbf{N}$ denote the set of noninferior objective vectors i.e.

$$\mathbf{N} = \{\mathbf{Z}(\mathbf{x}) \mid \mathbf{x} \in X^N\}$$

These sets are related as follows:

$$X^M \subseteq X^N \subseteq X^P \subseteq X^S$$

$$X^E \subseteq X^P$$

$$Q^M \subseteq Q^E \subseteq \mathbf{N} \subseteq Q^P$$

$$z_\omega^{min} \leq z_\omega^{max} \leq z_\omega^* \qquad \text{for all } \omega \in \Omega$$

4.6 Assumptions

The principal assumption required for the application of noninferior set scenario analysis is that the decision maker has a strictly increasing preference structure for all scenarios. That is, for a maximising problem, the decision maker will always want to increase the objective function value for every scenario, irrespective of the objective function values of the other scenarios. This assumption is required to support the definition of noninferiority given above. If there is a level of attainment for one of the scenario objectives beyond which the decision maker is indifferent to increases in value, or even prefers the value to decrease, then the definition of noninferiority given above can no longer be used.

Our definition of noninferiority also assumes that the decision maker's preference structure under each scenario is independent of the value of the objectives attained under the other scenarios. This means that, whatever the objective values under the scenarios, the decision maker would prefer an increase in the value of any of them, to holding them all at the same value, or decreasing any of them. This does not mean that the decision maker will never be willing to accept a decrease in the objective value under one scenario in order to get an increase in the objective value under another. Neither does it mean that the decision maker is indifferent to which scenario has its objective value improved, but it does mean that more is always better, wherever he or she can get it.

There is an important difference between decision making under uncertainty, and the deterministic, multiobjective optimisation problem that noninferior set scenario analysis is based on. In decision making under uncertainty, because only one scenario will actually occur, the decision maker will observe only one outcome, and only one objective function value. In contrast, in deterministic multiobjective decision making, the decision maker will observe every objective function value. This means that the decision maker gets a "whole package" under deterministic multiobjective optimisation, and a poor value for one objective can be compensated for by a good value for another. When decisions are taken under uncertainty, a poor result under the observed scenario is all that the decision maker gets. Knowledge that a better result would have been obtained had a different scenario occurred is little compensation. Further, the decision maker will feel regret if a different decision, that would have performed better under the observed scenario, was available, but not implemented. This will be especially true if the decision maker gambled on the observed scenario not occurring and traded off against it for a good result under a scenario that did not eventuate. On the other hand, of course, a decision maker may gamble and win. Decision making under uncertainty, and so Noninferior Set Scenario Analysis, involves considerations of risk, and attitudes to risk, that are not part of deterministic multiobjective decision making.

4.7 Problems With Two Scenarios

For simplicity of exposition, we will first discuss the formulation and solution method for the two scenario problem. In Chapter 5 we will expand it to the three scenario case.

Consider the following problem with two scenarios:

$$\begin{array}{ll} \text{maximise} & \left\{ \begin{array}{l} z_1 = (\mathbf{c}_0 + \mathbf{c}_1)\mathbf{x} + \mathbf{q}_1\mathbf{y}_1 \mid A\mathbf{x} = \mathbf{b}_0\,;\; B_1\mathbf{y}_1 + T_1\mathbf{x} = \mathbf{b}_1 \\ z_2 = (\mathbf{c}_0 + \mathbf{c}_2)\mathbf{x} + \mathbf{q}_2\mathbf{y}_2 \mid A\mathbf{x} = \mathbf{b}_0\,;\; B_2\mathbf{y}_2 + T_2\mathbf{x} = \mathbf{b}_2 \end{array} \right\} \\ \text{subject to} & \mathbf{y}_1 \geq 0,\;\; \mathbf{y}_2 \geq 0,\;\; \mathbf{x} \geq 0 \end{array} \tag{4.8}$$

The first step in the analysis is to find the scenario-maximal decision vectors, and the ideal objective vector. This is done by solving the two problems (4.9).

Scenario 1

$$\text{maximise} \quad z_1 = (\mathbf{c}_0 + \mathbf{c}_1)\mathbf{x} + \mathbf{q}_1\mathbf{y}_1 \tag{4.9a}$$

$$\text{subject to} \quad A\mathbf{x} = \mathbf{b}_0 \tag{4.9b}$$

$$B_1\mathbf{y}_1 + T_1\mathbf{x} = \mathbf{b}_1 \tag{4.9c}$$

$$B_2\mathbf{y}_2 + T_2\mathbf{x} = \mathbf{b}_2 \tag{4.9d}$$

$$\mathbf{x} \geq 0, \quad \mathbf{y}_1 \geq 0, \quad \mathbf{y}_2 \geq 0 \tag{4.9e}$$

Scenario 2

$$\text{maximise} \quad z_2 = (\mathbf{c}_0 + \mathbf{c}_2)\mathbf{x} + \mathbf{q}_2\mathbf{y}_2 \tag{4.9a}$$

$$\text{subject to} \quad A\mathbf{x} = \mathbf{b}_0 \tag{4.9b}$$

$$B_2\mathbf{y}_2 + T_2\mathbf{x} = \mathbf{b}_2 \tag{4.9c}$$

$$B_1\mathbf{y}_1 + T_1\mathbf{x} = \mathbf{b}_1 \tag{4.9d}$$

$$\mathbf{x} \geq 0, \quad \mathbf{y}_2 \geq 0, \quad \mathbf{y}_1 \geq 0 \tag{4.9e}$$

These two problems produce the objective values, z_1^{max} and z_2^{max}, respectively. The second term for each of the objective vectors are found by solving the two problems:

Scenario 1

$$\text{maximise} \quad z_1(z_2^{max}) = (\mathbf{c}_0 + \mathbf{c}_1)\mathbf{x} + \mathbf{q}_1\mathbf{y}_1 \tag{4.10a}$$

$$\text{subject to} \quad A\mathbf{x} = \mathbf{b}_0 \tag{4.10b}$$

$$B_1\mathbf{y}_1 + T_1\mathbf{x} = \mathbf{b}_1 \tag{4.10c}$$

$$B_2\mathbf{y}_2 + T_2\mathbf{x} = \mathbf{b}_2 \tag{4.10d}$$

$$(\mathbf{c}_0 + \mathbf{c}_2)\mathbf{x} + \mathbf{q}_2\mathbf{y}_2 \geq z_2^{max} \tag{4.10e}$$

$$\mathbf{x} \geq 0, \quad \mathbf{y}_1 \geq 0, \quad \mathbf{y}_2 \geq 0 \tag{4.10f}$$

Scenario 2

$$\text{maximise} \quad z_2(z_1^{max}) = (\mathbf{c}_0 + \mathbf{c}_2)\mathbf{x} + \mathbf{q}_2\mathbf{y}_2 \tag{4.10a}$$

$$\text{subject to} \quad A\mathbf{x} = \mathbf{b}_0 \tag{4.10b}$$

$$B_2\mathbf{y}_2 + T_2\mathbf{x} = \mathbf{b}_2 \tag{4.10c}$$

$$B_1\mathbf{y}_1 + T_1\mathbf{x} = \mathbf{b}_1 \tag{4.10d}$$

$$(\mathbf{c}_0 + \mathbf{c}_1)\mathbf{x} + \mathbf{q}_1\mathbf{y}_1 \geq z_1^{max} \tag{4.10e}$$

$$\mathbf{x} \geq 0, \quad \mathbf{y}_2 \geq 0, \quad \mathbf{y}_1 \geq 0 \tag{4.10f}$$

where: $z_i(z_j^{max})$ denotes that the objective function for scenario i is optimised subject to scenario j achieving its scenario-maximal objective function value.

Rather than fixing the stage one decision vector to be the scenario-maximal stage one decision vector of the other scenario, constraint (4.10e) requires that the scenario-maximal objective function value for the other scenario be attained. This means that if problem j has alternative optima, the one that performs best under scenario i will be found. This ensures that the resulting criterion vectors will be strongly noninferior.

We now have the two objective vectors $(z_1^*(z_2^{max}), z_2^{max})$ and $(z_1^{max}, z_2^*(z_1^{max}))$. In the two-scenario case, the minimum objective function values over the noninferior set, z_1^{min} and z_2^{min} occur when the objective function value for the other scenario is at its scenario-maximal value. Thus the noninferior points in objective space that bound the noninferior set are the two points, (z_1^{min}, z_2^{max}) and (z_1^{max}, z_2^{min}), where

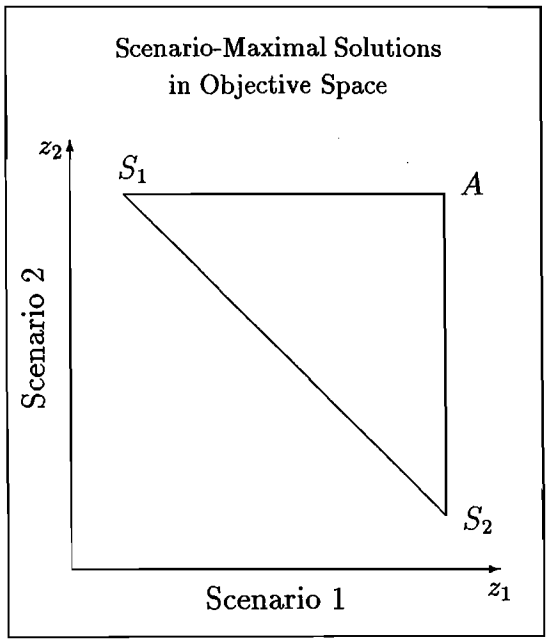

Figure 4.2:
The Initial Zone of Noninferiority
Points S_1 and S_2 are the extreme points corresponding to the scenario-maximal decisions. The noninferior set must lie within the triangle S_1AS_2.

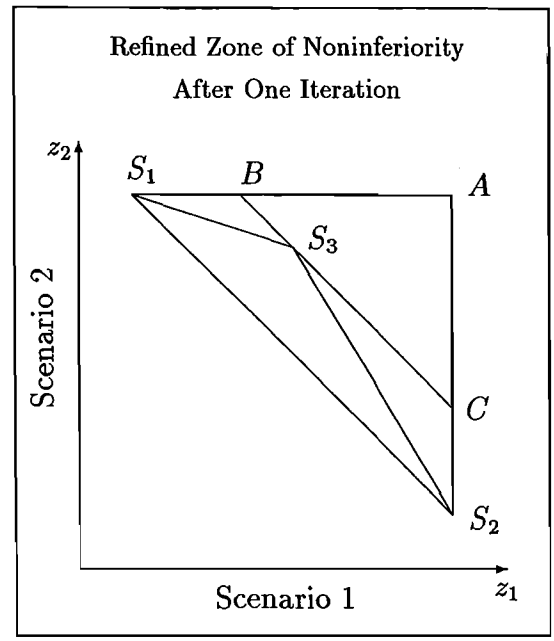

Figure 4.3:
A Refined Zone of Noninferiority
The line segments $S_1S_3S_2$ form an approximation of the noninferior solution set. The exact set must lie within the two triangles S_1BS_3 and S_3CS_2

$z_1^{min} = z_1^*(z_2^{max})$ and $z_2^{min} = z_2^*(z_1^{max})$. This result is particular to the two-scenario case, and does not apply to problems with more than two scenarios and higher dimensioned objective spaces. In problems with objective function spaces of more than two dimensions, the minimum scenario objective values over the noninferior set are not normally associated with the scenario-maximal solutions.

We use the NISE method of Cohon (1978) (see also Cohon et al. 1979) to find an approximation to the noninferior set in objective space. Because the NISE algorithm is described in detail in Cohon (1978) and Cohon et al. (1979), we only summarise it here. The method proceeds as follows.

We now have the two noninferior points, $S_1 = (z_1^{min}, z_2^{max})$ and $S_2 = (z_1^{max}, z_2^{min})$ in a two dimensional objective space. These points are shown in Figure 4.2. Because it is impossible for any problem-feasible decision to have a better z_1 value than z_1^{max}, or a better z_2 value than z_2^{max}, the lines S_1A and AS_2 form an upper bound on the noninferior set. Because the problem is convex, the line S_1S_2 is a lower bound on the noninferior set. The region S_1AS_2 is referred to as a *zone of noninferiority*, being the only region in objective space in which the actual noninferior set may lie.

If the lower bound S_1S_2 is used as an approximation for the noninferior set, then the greatest possible error between any point in the approximation and the true noninferior set is the maximum distance between line S_1S_2 and the upper bound. In Figure 4.2, this is the distance between line S_1S_2 and point A.

The NISE algorithm searches for a new noninferior solution in a direction at right angles to the lower bound by forming the weighted problem:

$$\text{maximise} \qquad z = w_1 z_1 + w_2 z_2 \tag{4.11a}$$

$$\text{subject to} \qquad A\mathbf{x} = \mathbf{b}_0 \tag{4.11b}$$

$$B_1\mathbf{y}_1 + T_1\mathbf{x} = \mathbf{b}_1 \tag{4.11c}$$

$$B_2\mathbf{y}_2 + T_2\mathbf{x} = \mathbf{b}_2 \tag{4.11d}$$

$$\mathbf{x} \geq 0 \;\; \mathbf{y}_1 \geq 0 \;\; \mathbf{y}_2 \geq 0 \tag{4.11e}$$

where: $-w_1/w_2$ = the slope of line S_1S_2

The new point is used to produce a better estimation of the bounds. In Figure 4.3, the criterion vector of the new solution is point S_3. Line BC, parallel to line S_1S_2, is a new segment of the upper bound. By convexity, lines S_1S_3 and S_3S_2 form a new lower bound on the noninferior set, and they are used as the new approximation of the noninferior set. The maximum possible error for line segment S_1S_3 is the distance between S_1S_3 and point B, and the greatest possible error for S_3S_2 is the distance between S_3S_2 and point C. Problem (4.11) is solved for whichever of the two segments of the lower bound, S_1S_3 and S_3S_2, has the greatest possible error. As before, the weights for the problem are derived from the slope of the selected line segment.

After each new noninferior point is added to the approximation the line segment with the largest value for the maximum possible error is used to find the next point. This means that the procedure concentrates on those parts of the approximation where the greatest improvements can be made. If, when problem (4.11) is solved for a line segment, the new point is an extreme point that has already been found, then the line segment is part of the exact noninferior set, and has a maximum possible error of zero.

The algorithm continues to subdivide the zone of noninferiority into successively

smaller triangles until the maximum possible error within every triangular section reaches a required tolerance. If the maximum permitted error is set to zero, the algorithm will find the exact noninferior set. However, if the exact noninferior set is required, every efficient solution must be found, and parametric programming is much more efficient for this purpose than the NISE algorithm.

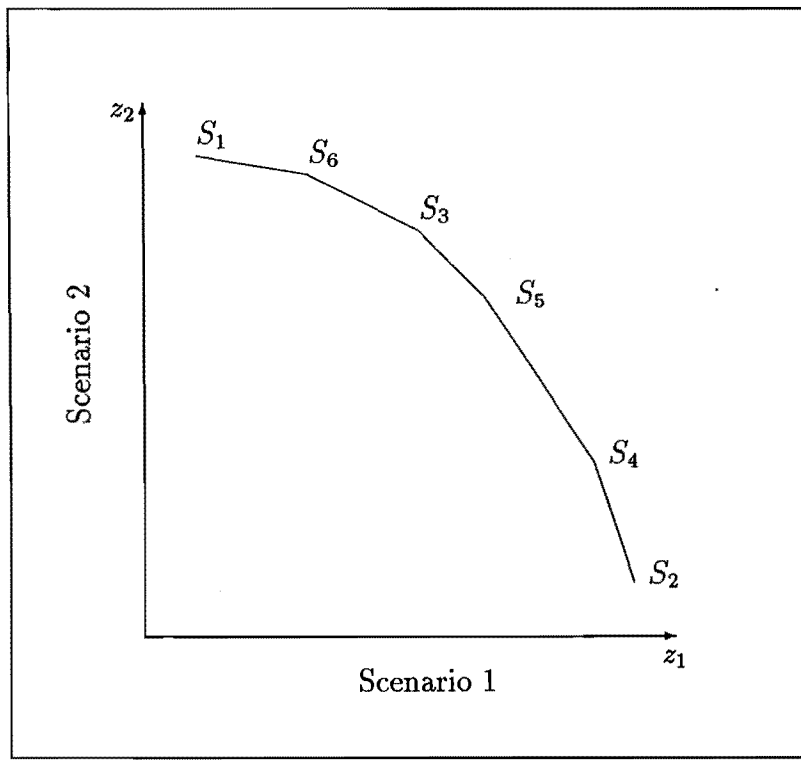

Figure 4.4: The final approximation to the Noninferior Set
The approximation to the noninferior set is defined by the six extreme points, S_1 to S_6. The points are indexed in the order in which they were found by the solution algorithm.

The final approximation to the noninferior set for this example is shown in Figure 4.4. The extreme points are numbered in the order in which they would be found by the algorithm. That is, in Figure 4.3, line segment S_3S_2 had the larger value for the maximum possible error and was used to derive the weights to find the next solution, and point S_4. Line segment S_3S_4 was then chosen to produce a solution, and finally segment S_1S_3 was used.

The noninferior set forms a frontier that has a typical knee shape, in which the trade-offs between the scenarios are extreme at the ends and more balanced in the middle. The decision maker can choose a point from this set to match her preferred

trade-off between the scenarios. Clearly she will not choose a point that lies below the frontier, because all such points are dominated by points on the frontier. If the noninferior set has been found exactly, then any point above the frontier is infeasible, and the decision maker must choose a point that lies on it. If the frontier is an approximation to the noninferior set, then there may be feasible points above it. Such points will dominate some of the line segments that form the frontier. If the decision maker is not willing to accept this possibility, then the NISE algorithm can be used to refine the approximation in the region of interest. For example, if her preferred point lies on line segment S_6S_3, in Figure 4.4, she can use the slope of S_6S_3 to find a new point. If the new point turns out to be either S_6 or S_3, then segment S_6S_3 is a segment of the exact noninferior set. If the decision maker obtains a new point, then she can continue to find new extreme points until the noninferior set has been described exactly in the region of interest.

The decision maker can further evaluate her options by considering the stage one decisions themselves, and how well they fit with concerns and objectives that were not included in the model. She should also look at the recourse decisions under each scenario to gain insights about the implications of her choice of stage one decision. The solution actually chosen by the decision maker will depend upon the problem, the decision maker's non-quantified objectives, and her attitude to risk.

4.7.1 Parametric Programming

The NISE approach analyses problem (4.11) by varying the relative weights placed the on variables z_1 and z_2. This could be achieved by holding one of w_1 or w_2 constant, and varying the other, using parametric programming. However, parametric programming has the drawback that it can only be used to find the noninferior set exactly, because it must find all of the extreme points. In contrast, the NISE approach is able to describe an approximation to the whole noninferior set with a subset of the extreme points. Further, if the search for an approximation is terminated by reaching an iteration limit, then parametric programming will describe part of the noninferior set exactly and provide no information about the rest of it. NISE on the other hand, will produce an approximation to the whole of the noninferior set.

More importantly, however, parametric programming becomes very difficult when

there are more than two scenarios. With three scenarios, one of w_1, w_2 or w_3 can be held constant, but two weights have to be varied simultaneously. As the number of scenarios increases, the difficulties, and the computational burden, rapidly increase.

Because parametric programming must find the noninferior set exactly, it produces results to an accuracy that is not appropriate to the problem. In strategic planning problems there is a great deal of uncertainty about the future, and the parameters of the problem cannot be specified to a high degree of accuracy. The magnitudes of the differences between adjacent noninferior solutions in an exact representation of the noninferior set will be smaller than the noise in the data, and thus meaningless. An exact representation of the noninferior set also gives a misleading sense of the accuracy of the results, and the decision maker runs the risk of wasting time choosing between alternatives that should be considered to be the same decision. The excessive level of detail will also obscure the main issues and make it difficult for the decision maker to gain insights about, and understanding of, the underlying problem.

4.7.2 Interpreting the Weights as Probabilities

The noninferior set can be interpreted to provide solutions to the stochastic optimisation problem (4.5). If the scenario objective functions can be combined into a meaningful expected value function, then the solution to the stochastic optimisation problem, for any pair of scenario probabilities, can be read from the noninferior set.

Consider problem (4.11), and the noninferior set of Figure 4.4. If this problem is viewed as a stochastic optimisation problem, with scenario probabilities w_1 and w_2, then the optimal solution for any probability pair can be read from Figure 4.4. If this interpretation is to be made, the weights should be normalised to lie in the range [0,1], and to sum to 1.

Each of the six extreme points of Figure 4.4 is associated with a range of scenario weights, and each of the five line segments on the frontier S_1 to S_2, has an associated pair of scenario weights $(w_1, w_2)_{ij}$, where i and j are the subscripts of two adjacent points. The ratio of the weights along one of the line segments is equal to the negative of the slope of the line. That is, if m_{ij} is the slope of line segment ij, then $(w_1)_{ij}/(w_2)_{ij} = -m_{ij}$.

If we interpret these weights as probabilities, then extreme point S_j is the optimal solution to problem (4.11) for scenario probabilities in the range $(w_1, w_2)_{ij}$ to $(w_1, w_2)_{jk}$, where ijk are the indices of three adjacent extreme points. The expected value of point S_j varies from $(w_1, w_2)_{ij}(z_1, z_2)_j$ to $(w_1, w_2)_{jk}(z_1, z_2)_j$.

All points on line segment ij are alternative optima for scenario weights $(w_1, w_2)_{ij}$, and the expected value is constant along the line segment with value $(w_1, w_2)_{ij}(z_1, z_2)_i = (w_1, w_2)_{ij}(z_1, z_2)_j$. In Figure 4.4, extreme point S_1 is optimal for the range $\left[(0,1)\right.$ to $\left.(w_1, w_2)_{16}\right]$, and S_2 is optimal for the range $\left[(w_1, w_2)_{42}\right.$ to $\left.(1,0)\right]$. Figure 4.5 shows the range of probabilities for which the extreme points in Figure 4.4 are optimal.

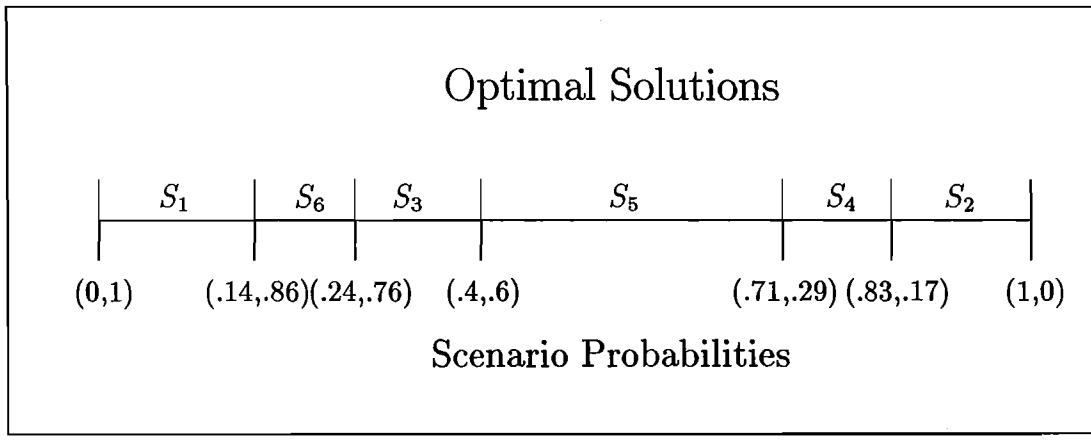

Figure 4.5: The Optimal Solutions Along the Probability Line
As the probabilities of the scenarios change from certainty for scenario 2, to certainty for scenario 1, the optimal solution changes from extreme point S_1 to S_6 to S_3 to S_5 to S_4 to S_2.

This interpretation of the weights as probabilities means that the decision maker can also evaluate a noninferior solution by considering the range of scenario probabilities over which it is the optimal solution to the expected value problem. Used in this fashion the noninferior set provides full sensitivity analysis on the scenario probabilities.

The NISE method can be directed to find the noninferior set over a particular range of weights, rather than finding the whole set. This permits the decision maker to find the subset of noninferior solutions that apply to a particular range of probabilities. For example, before solving problem 4.11 the decision maker could have specified that he or she was only interested in solutions that were optimal within the probability range $(0.2, 0.8)$ to $(0.8, 0.2)$, in which case the NISE algorithm would have found the approximation described by points $S_6 S_3 S_5 S_4$.

4.8 Summary

In this chapter we have presented the theoretical basis of Noninferior Set Scenario Analysis. In Sections 4.2 and 4.3 we described the problem situation that NSSA is designed to analyse, and how this problem can be formulated as a scenario analysis problem, and as a two-stage stochastic optimisation problem. In Section 4.4 we showed how this two-stage problem can be considered to have multiple objectives, one for each scenario, and we formulated the problem as a multiobjective optimisation problem. In Sections 4.5 and 4.6 we presented definitions and notation, and discussed the assumptions about the decision maker that are required for this approach to be applicable. In Section 4.7 we described the approach as it applies to problems with two scenarios, and showed how it is used to identify and describe the nondominated alternatives that are available to the decision maker. Decision makers can choose between these alternatives to find a decision that best matches their attitudes to risk, and which accommodates concerns that could not be included in the mathematical programming representation of the problem. We then show how the results of the analysis can be interpreted to find the optimal expected value solution for the problem for any pair of scenario probabilities.

The contribution to the literature of this chapter is the conceptualisation of the scenario analysis problem as a multiobjective problem, and the new view of decision making under uncertainty that this makes possible. The work described in this chapter uses the set generation technique of Cohon (1978) and Cohon et al. (1979) without modification, and it should be made clear that the algorithm itself is not being presented as new work.

Chapter 5

Problems With Three Scenarios

5.1 Introduction

In this chapter we extend our approach to problems with three scenarios by applying the XNISE algorithm of Solanki et al. (1993). This algorithm extends the bi-criterion algorithm of Cohon (1978) and Cohon et al. (1979) that was used in Chapter 4 to problems of higher demensionality. We have specialised this algorithm to take advantage of working in just three dimensions, and we have implemented the algorithm using code written in "AMPL" (*A Modelling Language for Mathematical Programming*).

Unfortunately, the increase from two scenarios to three leads to much greater complexity in the solution method. The noninferior set now consists of connected edges and 2-D faces in a three dimensional objective space. The two dimensional approach of pushing out the efficient frontier by maximising at right angles to each segment of the current lower bound can no longer be relied upon to find all of the noninferior set.

The problem consists of three scenarios:

$$\text{maximise} \quad \left\{ \begin{array}{l} z_1 = (\mathbf{c}_0 + \mathbf{c}_1)\mathbf{x}_1 + \mathbf{q}_1\mathbf{y}_1 \mid A\mathbf{x}_1 = \mathbf{b}_0 \,;\; B_1\mathbf{y}_1 + T_1\mathbf{x}_1 = \mathbf{b}_1 \\ z_2 = (\mathbf{c}_0 + \mathbf{c}_2)\mathbf{x}_2 + \mathbf{q}_2\mathbf{y}_2 \mid A\mathbf{x}_2 = \mathbf{b}_0 \,;\; B_2\mathbf{y}_2 + T_2\mathbf{x}_2 = \mathbf{b}_2 \\ z_3 = (\mathbf{c}_0 + \mathbf{c}_3)\mathbf{x}_3 + \mathbf{q}_3\mathbf{y}_3 \mid A\mathbf{x}_3 = \mathbf{b}_0 \,;\; B_3\mathbf{y}_3 + T_3\mathbf{x}_3 = \mathbf{b}_3 \end{array} \right\} \qquad (5.1\,\text{a})$$

$$\text{subject to} \qquad \mathbf{y}_\omega \geq 0,\ \mathbf{x} \geq 0 \qquad (5.1\,\text{b})$$

The problem is formulated as problem (5.2) in which the nonanticipativity restriction is enforced by having a single stage one decision vector, and all of the constraints for all of the scenarios are gathered into a single constraint set to ensure that the stage one decision is feasible under all scenarios.

Constraint (5.2 d) is included in the formulation to enable the decision maker to set minimum acceptable objective function values for the scenarios. The inclusion of constraint (5.2 d) avoids generating parts of the noninferior set that are known, *a priori* to be unacceptable to the decision maker.

$$\begin{aligned}
&\text{maximise} && V = w_1 z_1 + w_2 z_2 + w_3 z_3 && && (5.2\,\text{a})\\
&\text{subject to} && z_\omega = (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x} + \mathbf{q}_\omega \mathbf{y}_\omega && \forall\ \omega \in \Omega &\\
& && A\mathbf{x} = \mathbf{b}_0 && && (5.2\,\text{b})\\
& && B_\omega \mathbf{y}_\omega + T_\omega \mathbf{x} = \mathbf{b}_\omega && \forall\ \omega \in \Omega && (5.2\,\text{c})\\
& && z_\omega \geq z_\omega^{lb} && \forall\ \omega \in \Omega && (5.2\,\text{d})\\
& && \mathbf{y}_\omega \geq 0,\ \mathbf{x} \geq 0 && && (5.2\,\text{e})
\end{aligned}$$

where: $\mathbf{w}$ = weights placed on the scenarios

z_ω^{lb} = the minimum acceptable objective function value for scenario ω

As with the two-scenario problem, the method starts by finding the scenario-maximal solutions. If these three stage one solutions are the same, then the three scenarios have the same maximal stage one decision, and the problem is solved. In general, this will not be the case, and (assuming that the three scenario-maximal criterion vectors are not in the same straight line) the first approximation to the noninferior set is the plane defined by these three points. This plane will be referred to as the *primary plane.*

Two complications arise in the 3-D problem which do not occur in the 2-D problem. Firstly, some of the noninferior set may lie to the side the primary plane, rather than above it (Cohon 1978, Gero 1985, Solanki et al. 1993). This means that search directions must be used in which some of the elements of the direction vector are negative, and these searches may lead to inferior solutions. Furthermore, it may be impossible to find the whole of the noninferior set without using some inferior points along the way, (Solanki et al. 1993).

This is illustrated in Figure 5.1. In this figure, A, B and C are the scenario-maximal extreme points, and ABC is the primary plane. Point D was found by

optimising with the weights of the outward normal of face ABC. Now let point E be inferior and point F be noninferior. The noninferior point F is said to lie to the side of the primary plane, and it is apparent from the figure that face BCD, which has a negative component along the z_2 axis, is the only face on the tetrahedron ABCD that could find point F. However, when the problem is optimised using the outward normal of face BCD to provide the weights, the inferior point E is the one that is found. Once point E has been found, face BCD is replaced by faces DCE, DBE and BCE. Point F can now be found by optimising with the weights of face DBE. This simple example illustrates how it may be necessary to use search directions with negative elements to find all of the noninferior extreme points, and that some inferior extreme points may have to be found in order to find all of the noninferior extreme points.

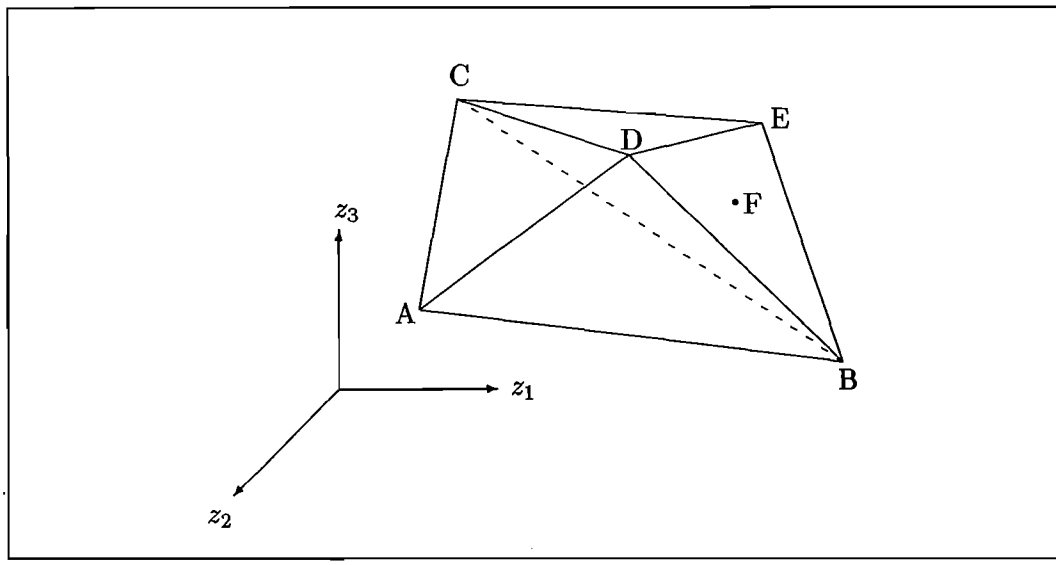

Figure 5.1: Negative Weights May be Required to Find the Noninferior Set
An example showing that it may be necessary to use negative weights to find all of the noninferior extreme points, and that inferior extreme points may have to generated along the way.

The second complication is that, whereas with two scenarios the scenario-maximal points also give the minimum objective function values over the noninferior set, this is not the case for problems with more than two objectives. Typically, the non-inferior points associated with the minimum objective function values are not the scenario-maximal points, and finding the minimum objective function values over the noninferior set is a non-trivial problem. This means that a lower bound for

the noninferior set is not readily available. Reeves & Reid (1988) investigated the differences between the subset of efficient solutions bounded by the minimum values associated with the maximal points, and the complete efficient set. They report that, for the sample problems tested, the increase in the range of objective function values associated with going from the subset to the full efficient set was not large. The increases observed varied from 3.9% to 7.5%. This suggests that the approach used in our work of using the minimum scenario objective function values associated with the scenario-maximal points as the lower bounds on the noninferior set will not seriously reduce the number of alternatives reported to the decision maker. Furthermore, it would seem unlikely that a decision maker who is not an extreme risk taker would choose any solution that produces the worst outcome for one of the scenarios.

5.2 Further Definitions and Notation

This section continues from Section 4.5 and uses terms and notation presented in that section.

Definition 5.1 *The convex hull of a set of extreme criterion vectors, U (U is a subset of Q^P), is defined as the set of all convex combinations of the vectors in U. The convex hull of U is denoted by CH(U).*

Definition 5.2 *A plane in a 3-D objective space is defined as the set of criterion vectors* $\mathbf{z} = H(\mathbf{w}, d)$ *where:*

$$H(\mathbf{w}, d) = \{\mathbf{Z} \mid w_1 z_1 + w_2 z_2 + w_3 z_3 = d \text{ and at least one of } w_i \neq 0\}$$

Definition 5.3 *The point* $\mathbf{Z}^k = (z_1^k, z_2^k, z_3^k)$, *is said to be* beyond *the plane* $H(\mathbf{w}, d)$ *if*

$$w_1 z_1^k + w_2 z_2^k + w_3 z_3^k > d$$

The point $\mathbf{Z}^k = (z_1^k, z_2^k, z_3^k)$, *is said to be* beneath *the plane* $H(\mathbf{w}, d)$, *if*

$$w_1 z_1^k + w_2 z_2^k + w_3 z_3^k \leq d$$

Clearly, whether a point is *beneath* or *beyond* with respect to plane $H(\mathbf{w}, d)$ depends on the direction given to vector $\mathbf{w}$.

Definition 5.4 *Let CH(U) be the convex hull of U where U is a subset of Q^P. Let V be a subset of U. Then $H(\mathbf{w}, d)$ is a supporting plane at V (i.e. $H(\mathbf{w}, d)$ supports CH(U) at V) if and only if*

(a) V is contained in $H(\mathbf{w}, d)$, and

(b) $\sum_{i=1}^{3} w_i z_i \leq d$ for all $\mathbf{Z} \in \mathrm{CH}(U)$

The vector $\mathbf{w} = (w_1, w_2, w_3)$ is termed the outward normal of the supporting plane $H(\mathbf{w}, d)$. An criterion vector $\mathbf{Z}$ satisfying the condition $\sum_{i=1}^{3} w_i z_i > d$ is necessarily out of the convex hull $\mathrm{CH}(U)$.

The convex hull $\mathrm{CH}(U)$ defines an inner bound of the noninferior set, $\mathbf{N}$, because points in the interior of the hull are dominated by one or more of the points on its surface.

Definition 5.5 *Let HUZ be the set of criterion vectors in the polytope that forms an outer bound on the noninferior set, $\mathbf{N}$, (that is, $\mathbf{N}$ cannot be outside HUZ), and let HU be the set of planar polygons that forms the surface of HUZ.*

Let $U \in Q^P$ be a set of extreme points in objective space.

Let a point $\mathbf{Z}^k \in U$ be (z_1^k, z_2^k, z_3^k)

Let a plane that passes through $\mathbf{Z}^k$, be $H(\mathbf{w}, d)$, such that $\mathbf{w}\mathbf{Z}^i \leq d$ for all $\mathbf{Z}^i \in Q^P$ i.e. all feasible points are in the half space $\mathbf{w}\mathbf{Z} \leq d$

Because $\mathbf{w}\mathbf{Z} \leq d$ for all feasible points, $\mathbf{w}\mathbf{Z} \leq d$ holds for all $\mathbf{Z} \in \mathrm{CH}(U)$.

Further, $\mathbf{Z}^k \in U$, so $H(\mathbf{w}, d)$ supports $\mathrm{CH}(U)$ at $\mathbf{Z}^k$.

When any feasible extreme point is added to U, updating it to set U', $H(\mathbf{w}, d)$ will also support $\mathrm{CH}(U')$, because all feasible points are in the half space $\mathbf{w}\mathbf{Z}$.

Because all feasible points are in the half space $\mathbf{w}\mathbf{Z} \leq d$, all of the noninferior points must also be in that half space.

Select one such plane for each $\mathbf{Z}^k \in U$, and denote this set of planes as HU

The intersection of the half spaces defined by HU defines the convex polytope, HUZ, which is an outer bound on $\mathbf{N}$. In other words, all noninferior points lie within (or on the surface of) the polytope HUZ.

That is, all points in the noninferior set must lie *beneath* the planes in HU.

Theorem 5.1 *A criterion vector* $\mathbf{Z}^k \in Q^P$ *is noninferior if and only if there exists a plane* $H(\mathbf{w}^k, d^k)$ *that supports* $CH(Q^P)$ *at* $\mathbf{Z}^k$ *and whose outward normal* $\mathbf{w}^k$ *is strictly positive i.e.* $w_i^k > 0$ *for* $i = 1, 2, 3$.

This theorem applies to linear problems and follows exactly from theorem 4.22 given in Chankong & Haimes (1983), page 154. A number of proofs exist, (see for example Isermann 1974).

Let U be a set of extreme points in objective space. The convex hull, CH(U), of these points forms an inner bound on $\mathbf{N}$. Each point, $\mathbf{Z}^k \in U$ is the solution to problem (5.2) for some weighting of the scenarios, $\mathbf{w}^k$. Associated with each $\mathbf{Z}^k$ is a plane $H(\mathbf{w}^k, d^k)$ which is a member of the set of outer bounding planes, HU. The noninferior set in objective space lies between the inner bound, CH(U), and the outer bound, HUZ. The inner and outer bounds touch at each $\mathbf{Z}^k \in U$. Although all points in U are extreme points, they are not all noninferior points.

The approach used by the XNISE algorithm is to progressively add extreme points to the set U, and so reduce the distance between the inner and outer bounds. Each new point expands the inner bound, and contracts the outer bound. When the distances between the bounds are sufficiently small, the noninferior set is assumed to lie on the surface of CH(U). Although the approximation lies on the surface of CH(U), it is important to recognise that it is not the entire surface, and that at least one face of CH(U) must be inferior.

The faces on the surface of CH(U) that are noninferior with respect to the points in CH(U) are then identified, and these form the approximation. Thus the approach proceeds in two stages. In the first stage extreme points are added until the bounds are sufficiently close together. In the second stage the approximation is formed from the faces and edges on the surface of the inner bound, CH(U), that are noninferior with respect to the points in CH(U).

5.3 The NSSA Algorithm With Three Scenarios

5.3.1 Initialisation

The algorithm must start by finding a starting inner bound, and a starting outer bound, which means that four distinct points in objective space are required. It seems natural to start by finding the three scenario-maximal criterion vectors, because the decision maker can be expected to be interested in knowing about them anyway. Also, for each of these points, the plane at right angles to the axis of the scenario being maximised is known to be beyond the noninferior set. (The point was found by maximising in the direction of that axis, so no feasible point can lie beyond it.) Thus these three points provide three of the planes needed to form the outer bounding polytope, HUZ. These three points lie in a plane (the primary plane), and a fourth point can be found by maximising at right angles to that plane. This fourth point is used to create a tetrahedron that is used as the starting inner bound for the noninferior set.

The algorithm starts by finding the three scenario-maximal criterion vectors, and thus the primary plane. Each scenario-maximal point is found by a pair of optimisations. First, problem (5.2) is solved with the weight set to a positive number for one scenario, say scenario i (we use 1, but any positive number can be used), and 0 for the others. This produces the scenario-maximal objective function value for scenario i, z_i^{max}, and constraint $z_i \geq z_i^{max}$ is appended to the problem. The remaining two scenarios must now be weighted to find the other coordinates for this scenario-maximal criterion vector. If both scenarios are given positive weights, then the remaining coordinates can be found by solving the problem one more time. If there is only one feasible point in objective space for which $z_i = z_i^{max}$, then any choice of positive weights will find that point. However, if there is a face, or an edge, over which $z_i = z_i^{max}$, then there will be a noninferior set in two dimensions, consisting of one, or more, connected edges. Different choices of weights will find different extreme points from that noninferior set, and the decision maker may wish to choose the weights to influence which point is found.

A goal programming approach could be used in which the weight for one of the remaining two scenarios would be made positive while leaving the weight for the third at zero, and the problem solved. The problem would then be optimised for the third scenario, with the objective function values of the first and second

scenarios fixed. This could be repeated with the third scenario weighted before the second scenario, to find the other end of the two-scenario noninferior set. This approach would have the advantage of providing better estimates for the minimum objective function values over the noninferior set, than those produced by weighting both scenarios together, but at the cost of additional computational effort. It also provides information about the regions of the noninferior set around the scenario-maximal solutions. However, if the decision maker is unlikely, in the end, to choose a solution that is close to the scenario-maximal value for that scenario, then the additional information obtained will be of little value. In addition, if the extreme points of any of these two-scenario noninferior sets are significantly different, they will eventually be found by the algorithm anyway.

If there is no reason to favour one of the remaining two scenarios over the other, then they can be weighted equally, or the weights chosen to scale the two objectives. For example, if the objective function values of the second scenario were in hundreds of thousands, and of the third in hundreds, then the use of equal weights would give similar results to preemptive goal programming with the second scenario weighted first. In such a case, weights of $w_2 = 1$ and $w_3 = 1000$ could be used to scale the problem. Once the weights have been chosen, the problem (5.2) is solved to find the other coordinates of the scenario-maximal point.

This procedure is carried out for each scenario in turn to find the coordinates in objective space of the three scenario-maximal extreme points.

On viewing these solutions, the decision maker may wish to specify, or revise, the minimum acceptable objective function values, z_ω^{lb}. If z_ω^{lb} is revised to a value that is larger than the objective function value attained by that scenario at one of the scenario-maximal points, then problem (5.2) should be solved again using the new bounds.

Values for the minimum objective function values over the noninferior set must also be determined. These are required to ensure that, when the weighting vector is not strictly positive, the resulting extreme points are not too far from the noninferior set. As already mentioned, these values do not appear in the scenario-maximal criterion vectors when there are more than two scenarios. Where the decision maker has specified values for z_ω^{lb}, then these are the correct values for z_ω^{min}, because any smaller value is infeasible, and so cannot be within the noninferior set. If values for z_ω^{lb} have not been specified, then the minimum values over the scenario-maximal

criterion vectors may be used as surrogates.

The three scenario-maximal objective function values are used to define three planes to form part of the initial outer bounding set, HU. For scenario ω, the plane is $z_\omega \leq z_\omega^{max}$ and is perpendicular to the z_ω axis. The minimum scenario objective function values over the scenario-maximal points are used to define three more planes, $z_\omega \geq z_\omega^{min}$, that also are perpendicular to the corresponding axes. Thus the surface of the initial outer bounding polytope, HUZ, is a six-sided, closed box.

Constraints are appended to problem (5.2) to ensure that any solution is within the outer bounding polytope. Thus problem (5.2) becomes:

$$\begin{aligned}
\text{maximise} \quad & V = w_1 z_1 + w_2 z_2 + w_3 z_3 && && (5.3\,\text{a})\\
\text{subject to} \quad & z_\omega = (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x} + \mathbf{q}_\omega \mathbf{y}_\omega && \forall \quad \omega \in \Omega && (5.3\,\text{b})\\
& A\mathbf{x} = \mathbf{b}_0 && && (5.3\,\text{c})\\
& B_\omega \mathbf{y}_\omega + T_\omega \mathbf{x} = \mathbf{b}_\omega && \forall \quad \omega \in \Omega && (5.3\,\text{d})\\
& w_1^u z_1 + w_2^u z_2 + w_2^u z_2 \leq V^u && \forall \quad u \in U && (5.3\,\text{e})\\
& \mathbf{y}_\omega \geq 0,\ \mathbf{x} \geq 0 && && (5.3\,\text{f})
\end{aligned}$$

where: $\mathbf{w}$ = weights placed on the scenarios

$\mathbf{w}^u$ = the outward normal of the supporting plane at extreme point u

$V^u = w_1^u z_1^u + w_2^u z_2^u + w_2^u z_2^u$

Constraint (5.3 e) ensures that the criterion vector found is within the outer bounding polytope. This prevents inferior extreme points from being too far from the noninferior set.

We now have the primary plane in objective space and the main algorithm could start from this point. However, it is easier to write the algorithm if the extreme points found so far describe a three dimensional convex hull rather than a two dimensional one. For this reason, the initialisation finds a fourth noninferior point and builds a three dimensional initial inner bound. The fourth point is found by solving problem (5.3) with the scenario weights set equal to the most positive normal vector of the primary plane. That is, $\mathbf{w}$ is chosen so that at least two of its terms are non-negative. If the fourth point lies in the primary plane, then the signs of the weights are reversed, and another solution found. If this point also lies in the primary plane, then the primary plane is the noninferior set, and the algorithm

terminates.

The set, U, is initialised as the four points found so far, and the HUZ is updated by adding the plane that supports CH(U) at the fourth point. This plane is $\mathbf{wZ} \leq V$, where $\mathbf{w}$ is the outward normal of the primary plane that was used to find the fourth point, and V is the optimal corresponding objective function value of problem (5.3). This plane is added to constraints (5.3e).

The three scenario-maximal points are known to be noninferior, but, if one or more of the terms of $\mathbf{w}$ were negative, the fourth point may be inferior. These four points are at the corners of a tetrahedron, which is the convex hull CH(U).

Finally, the maximum allowable error, *mae*, must be determined. The approach used by Solanki et al. (1993) is to set it to a fraction of the longest diagonal across the outer bound, HUZ. Clearly, any fraction could be used, but smaller maximum permitted errors require greater computational effort, while large fractions will find a less exact approximation. If mae = 0, then the noninferior set will be found exactly.

To facilitate data storage, and to simplify the logic of adding new points to U, the algorithm is written so that all faces on the surface of CH(U) are triangular, and have exactly three neighbours. However, it does not ensure that CH(U) is simplicial, because a vertex can have more than three faces incident on it. Even when adjacent faces are coplanar, they are maintained as separate faces so that all faces are triangular.

5.3.2 Data Storage

- The points in the set U are indexed, and stored with:

  Their coordinates in objective space.

  A flag set to show if they are known to be noninferior. A point is known to be noninferior if it was found using a strictly positive weighting vector.

- The faces of CH(U) are indexed and stored with the following information:

  The indices of the three corner points. These points are members of U.

  The equation of the outward normal, $H(w^f, d^f)$

  The indices of its three neighbouring faces,
  (The three faces that each share an edge with this face.)

- A supporting plane for each point $u \in U$, $H(\mathbf{w}^u, V^u)$. These planes form the set HU and define the outer bounding polytope, HUZ. The outward normal, $\mathbf{w}^u$, of the supporting plane at point u is the weighting vector that was used in problem (5.3) to find that point, and the constant term, V^u is the corresponding optimal weighted objective function value.

5.3.3 Procedure XNISE1

5.3.3.1 Select the Face with the Maximum Possible Error

The algorithm proceeds by finding the maximum possible error associated with each face, f, of CH(U). This is done by solving problem (5.4) for each face.

$$\text{maximise} \qquad z^H = w_1^f z_1 + w_2^f z_2 + w_3^f z_3 \tag{5.4 a}$$

$$\text{subject to} \qquad w_1^u z_1 + w_2^u z_2 + w_2^u z_2 \leq V^u \qquad \forall\ u \in U \tag{5.4 b}$$

where: $\mathbf{w}^f$ = the terms of the outward normal of the face being evaluated. For ease of error calculation, the weights are normalised so that: $(w_1^f)^2 + (w_2^f)^2 + (w_3^f)^2 = 1$

$\mathbf{w}^u$ = the outward normal of the supporting plane at point u

$V^u = \mathbf{w}^u \mathbf{z}^u$

$\mathbf{z}^u$ = coordinates of point u

Constraints (5.4 b) define the outer bounding polytope, HUZ.

For each face, f, of CH(U), problem (5.4) determines how far the face could be pushed out without leaving the upper bounding polytope, HUZ. This is the maximum possible distance that the face, f, can be from the true noninferior set in criterion space. As such it gives an upper bound on the error associated with using face f as part of the approximation to the noninferior set.

The distance between the point z^H and the plane $H(\mathbf{w}^f, d^f)$ is:

$$\frac{|w_1^f z_1 + w_2^f z_2 + w_3^f z_3 - d^f|}{\sqrt{(w_1^f)^2 + (w_2^f)^2 + (w_3^f)^2}}$$

Thus, because the weights of the face equation, $H(\mathbf{w}^f, d^f)$ have been normalised, the maximum error associated with face f is: $\delta^f = z^H - d^f$.

The face with the maximum error is selected. If this error is within the maximum allowable error, the approximation is complete, so STOP.

5.3.3.2 Find a New Extreme Point

The scenario weights are set to the outward normal of the selected face, and problem (5.3) is solved. As has already been observed, the CH(U) may have to be expanded in all directions for all noninferior points to be found, and faces with outward normals that are not strictly positive may have to be used. Constraints (5.3 e) keep any inferior extreme points that may be generated from being too far from **N**.

If one of the terms of **w** is zero, say w_i, then it may be possible to improve the objective function value for scenario i. To find this value, the objective function values for the other two scenarios are fixed, w_i is set to 1, and problem (5.3) is solved again.

If the new point is coplanar with the face used to find it, then no feasible points can be found by pushing that face out. This means that, not only does the face lie on the surface of CH(U), but it is also part of the outer bound HUZ. If the weighting vector was strictly positive, then the face is, in fact, part of the noninferior set. In any case its error has been shown to be zero. The face will not be selected again, because it now forms part of HUZ, and its outward normal, $H(\mathbf{w}, d)$, is used to create another constraint in the constraint set (5.4 b).

If the selected face has been shown to be part of HUZ, then the face with the next largest error is used to find a new extreme point.

5.3.3.3 Update the Approximation

The new point, call it H, is now added to the set of points, U, creating set U', and CH(U) is updated to become CH(U'). Because H lies outside CH(U), at least one of the faces on the surface of CH(U) will be interior to CH(U'), and must be discarded. Consider Figure 5.2. This is a view from outside CH(U), looking at it from an arbitrary direction. The extreme points A to G are members of U, with point A "in front", and point G "behind", in this view. The faces are numbered as shown, and the current list of faces is shown in Table 5.1. The new point, H, was found by pushing out face 2.

Point H is found to be *beyond* face 2, and *beneath* faces 1, 3, 4, 5, 6, 7, 8, 9 and

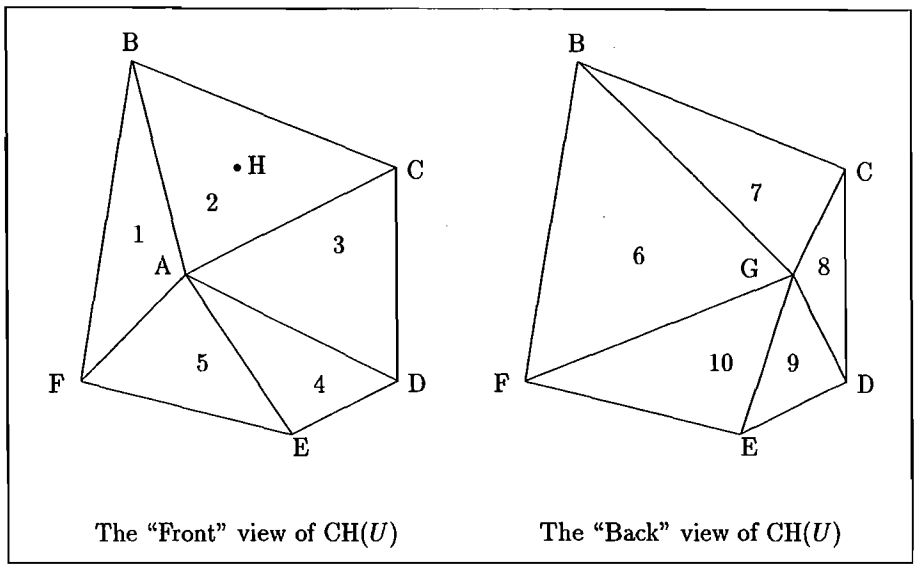

Figure 5.2:
The Convex Hull, Ch(U), of the Set of Extreme Points, U, Found So Far.

10. This can be visualised by extending the plane of face 3 (for example) over the top of face 2. Point H will be between this plane and face 2. This means that the points on face 2, with the exception of the edges AB, BC and CA, are in the interior of CH(U'). The surface of CH(U') is formed by replacing face 2 with the new faces HAB, HBC, and HCA. The new faces are numbered 2, 11 and 12 respectively. The new inner bound, CH(U') is shown in Figure 5.3. The list of faces is updated as shown in Table 5.2.

List of Faces in the "Front" View			List of Faces in the "Back" View		
Face	Vertices	Neighbours	Face	Vertices	Neighbours
2	ABC	3 1 7	7	GBC	8 6 2
3	ACD	4 2 8	8	GCD	9 7 3
4	ADE	5 3 9	9	GDE	10 8 4
5	AEF	1 4 10	10	GEF	6 9 5
1	AFB	2 5 6	6	GFB	7 10 1

Table 5.1: The List of Faces Corresponding to the CH(U) in Figure 5.2

The outward normals have to be found for each of the new faces. The equation of a face is found using the three corner points, but it is necessary to ensure that this equation is the outward normal. The old face 2 is known to be beneath the new faces, so its centroid is found, and the signs of the weights of the outward normals

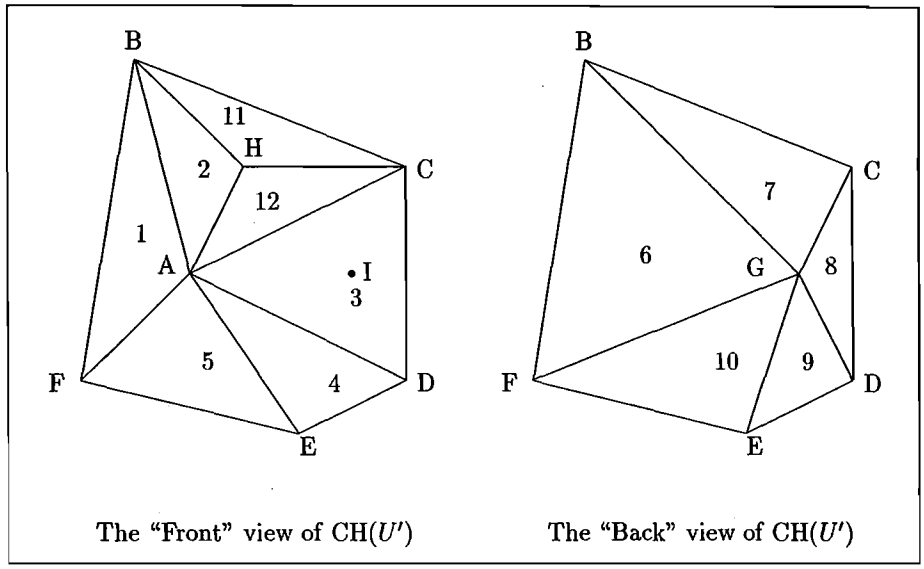

Figure 5.3: The Convex Hull, CH(U'), After the Addition of Point H to CH(U)

of the new faces are chosen so that the centroid of the old face 2 is *beneath* each of the new faces.

Finally, the outer bound, HUZ, is tightened by adding the objective function used to find point H, to the constraint set (5.4 b).

CH(U') is now fully described, and the algorithm calculates the maximum possible error for each face in CH(U').

The next iteration of XNISE1 will be described to illustrate how the inner bound is updated when the addition of a new extreme point could lead to the surface of the new set U'' being non-convex.

List of Faces in the "Front" View

Face	Vertices	Neighbours
11	HBC	12 2 7
12	HCA	3 2 11
2	HAB	11 12 1
3	ACD	4 12 8
4	ADE	5 3 9
5	AEF	1 4 10
1	AFB	2 5 6

List of Faces in the "Back" View

Face	Vertices	Neighbours
7	GBC	8 6 11
8	GCD	9 7 3
9	GDE	10 8 4
10	GEF	6 9 5
6	GFB	7 10 1

Table 5.2: The List of Faces Corresponding to the CH(U') in Figure 5.3

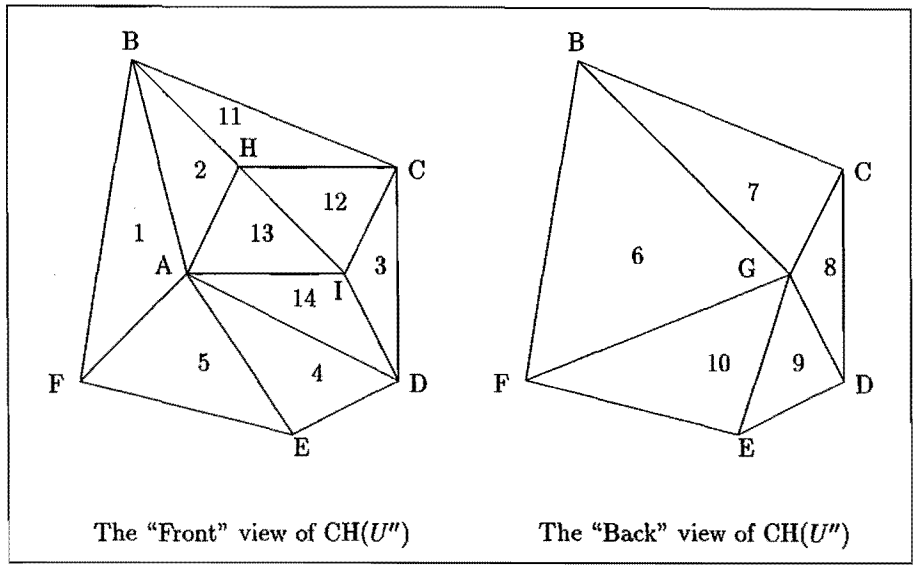

Figure 5.4: The Convex Hull, CH(U''), After the Relocation of Point H to CH(U')

Face 3 in CH(U'), see Figure 5.3, now has the greatest possible error, and its outward normal is used as the scenario weights to find a new extreme point. This new point, I is *beyond* faces 3 and 12, and coplanar with face 4. It is *beneath* all of the other faces. This means that faces 3 and 12 will be interior to the new inner bound, CH(U''). If face 3 were replaced by three new faces, as was done when H was added at the previous iteration, the surface of CH(U'') would not be convex. The edge AC would be at the bottom of a "valley" between points H and I. The required update is to replace faces 3 and 12 with four new faces, being IAH, IHC, ICD, and IDA.

The algorithm builds the new faces by identifying the edges that are between a face that the new point is beyond, and a face that the new point is beneath. The new faces are formed by using each of these edges as the base of a new triangular face, with the new point as the apex. Thus, CH(U'') is formed by replacing the two interior faces, 3 and 12, with the four new faces as shown in Figure 5.4. The updated list of faces is shown in Table 5.3. Point I was coplanar with face 4, so faces 4 and 14 lie in the same plane. However, they are retained as separate faces so that all faces are triangular.

Once the bounds have been tightened to the required extent, the actual approximation of N must be found from CH(U). This is carried out by procedure XNISE2.

List of Faces in the "Front" View				
Face	Vertices	Neighbours		
11	HBC	12	2	7
12	HCI	3	13	11
13	HIA	14	2	12
2	HAB	13	1	11
3	CID	14	12	8
14	AID	4	13	3
4	ADE	5	14	9
5	AEF	1	4	10
1	AFB	2	5	6

List of Faces in the "Back" View				
Face	Vertices	Neighbours		
7	GBC	8	6	11
8	GCD	9	7	3
9	GDE	10	8	4
10	GEF	6	9	5
6	GFB	7	10	1

Table 5.3: The List of Faces Corresponding to the CH(U'') in Figure 5.4

5.3.4 The XNISE1 Algorithm

Find the Scenario-Maximal Criterion Vectors, and the Initial Approximation to the Noninferior Set

Step 0 Get initial parameter settings:

- the fraction (mae_frac) of the longest diagonal of the initial approximation to use as the Maximum Allowable Error (MAE)
- the fraction (z_min_frac) of the minimum scenario objective function values over the three scenario-maximal criterion vectors to use to form the lower bounds (z_ω^{lb})

Step 1 Find the scenario-maximal solutions and the primary plane

For the three scenarios: $i = 1, 2, 3$

- set $w_1, w_2, w_3 = 0$
- set $w_i = 1$
- solve problem (5.2) and get the objective function value z_i^{max}
- append constraint $z_i \geq z_i^{max}$ to problem (5.2)
- set $w_1, w_2, w_3 = 1$
- solve problem (5.2)
- store the optimal criterion vector
- add it to the list of points known to be noninferior
- delete constraint $z_i \geq z_i^{max}$

End For

Step 2 Check that we have a plane

If the three points are the same point, then

- the problem has a single, optimal solution
- STOP

End If

If any two of the points are the same point, then

- try to find a unique third point
- set $w_1, w_2, w_3 = 1$
- solve problem (5.2)

If the new point is the same as one of the first three, then

- STOP
- otherwise, use the three unique points as the first three points

End If

End If

Step 3 Set the lower bounds on the scenario objective function values, z_ω^{lb}

For each scenario

- find the minimum objective function value over the scenario-maximal solutions
- multiply by the z_min_frac value entered at the start

End For

Step 4 Find a fourth criterion vector

- calculate the equation of the primary plane
- choose its direction to be positive in as many dimensions as possible
- solve problem (5.2) with this weighting vector

If the new point is in the primary plane, then

- multiply the weights by -1 to reverse the direction of the weighting vector
- solve problem (5.2) with this weighting vector

If this point is also in the primary plane, then

- the primary plane is the noninferior set
- STOP

End If

End If

Step 5 These four points characterise the initial approximation to the noninferior set

- calculate the longest diagonal across the tetrahedron
- use it to calculate the maximum allowable error (MAE)
- form the outward normals of the four faces
- flag the faces as not yet having maximum possible error (MPE) values
- initialise the outer bounding polytope, HUZ, as the constraints:

$$z_\omega \geq z_\omega^{lb} \qquad \forall\ \omega \in \Omega$$
$$z_\omega \leq z_\omega^{max} \qquad \forall\ \omega \in \Omega$$

Find New Criterion Vectors and Refine the Approximation

Step 6 Find the face with the largest maximum possible error (MPE)

For each face that does not have a current MPE

- calculate its MPE by solving problem (5.4)

End For

- select the face with the maximum MPE

If this MPE $\leq$ MAE, then

- the approximation has reached its required accuracy
- STOP

End If

- use the outward normal of the chosen face as the weighting vector, $\mathbf{w}^f$
- solve problem (5.3) with $\mathbf{w}^f$ to get criterion vector $\mathbf{z}^f$

If $\mathbf{w}^f$ has a zero term, then

- it may be possible to improve the corresponding objective function value, so:
- append constraint $z_i \geq z_i^f$ for the scenarios with $w_i \neq 0$
- set $w_i = 1$
- solve problem (5.3) to get a new $\mathbf{z}^f$
- delete constraints $z_i \geq z_i^f$

End If

If the new point is coplanar with the face, then

- the face is part of both the inner and outer bounding polytopes

- set its MPE to zero, and flag it as having a current MPE
- Goto **Step 6**

End If

- name the new point $\mathbf{z}^f$ as $\mathbf{v}$
- find the centroid of the chosen face, call it vector $\mathbf{c}^f$
- this point will be in the interior of CH(U')

Step 7 Update the outer bounding polytope:

- update problem (5.3) with the constraint:
  $\mathbf{w}^f\mathbf{z} \leq V$ where: $V = z^f$, the optimal objective function value of problem (5.3) for face f

This section updates the convex hull CH(U) to CH(U')

Step 8 Add the new criterion vector to the approximation

- Add the new vector, $\mathbf{v}$, to U to create U'

  If the weighting vector, $\mathbf{w}^f$, was strictly positive, then
  - $\mathbf{v}$ is noninferior
  - so add it to the list of points known to be noninferior

  End If

Step 9 Classify the faces of CH(U) as "beneath" or "beyond" with respect to $\mathbf{v}$

- "beneath" faces will be on the surface of CH(U')
- "beyond" faces will be interior to CH(U')

For for each face f in CH(U)

- let the outward normal of the face be $\mathbf{w}^f\mathbf{z} = d^f$

  If the dot product: $\mathbf{w}^f \cdot \mathbf{v} \leq d^f$, then
  - $\mathbf{v}$ is "beneath" face f
  - otherwise, $\mathbf{v}$ is "beyond" face f

  End If

End For

Step 10 Build the new convex hull CH(U')

- find the edges that are between a "beyond" face and a "beneath" face
- each of these edges will be the edge of a new face, with point $\mathbf{v}$ at the opposite corner

- name the new faces, record their vertices, and add them to the list of faces
- delete the "beyond" faces from the list of faces
- update the list of neighbouring faces for each face (both existing and new)
- calculate the equations of the new faces: $\mathbf{w}^f\mathbf{z} = d^f$
- use the interior point, c^f, to ensure that these equations are the outward normals:

  If the dot product: $\mathbf{w}^f \cdot \mathbf{c}^f > d^f$, then
  - the direction of the face equation is into CH(U')
  - so, multiply $\mathbf{w}^f$ by -1 to reverse the direction

  End If

Step 11 Find the next criterion vector to add to the approximation
- Goto **Step 6**

5.3.5 Procedure XNISE2

Once a set of extreme points has been found, the next step is to determine which of the faces and edges on the surface of their convex hull should, in fact, be included in the approximation of the noninferior set. The XNISE2 procedure identifies the faces that are noninferior with respect to the points in the convex hull. It also identifies any edges that are between two inferior faces, but are, themselves, noninferior

Normally some of the extreme points in U will be inferior, and at least one of the faces on the surface of CH(U), must be inferior. The XNISE2 procedure determines which faces and edges are noninferior with respect to the points in CH(U), and these faces are used as the approximation to the noninferior set.

This procedure considers each face in turn, and determines whether it is noninferior. If it is noninferior, it is added to the approximation. If not, its edges are tested to see if they are noninferior, and any noninferior edges are also added to the approximation. Unless the maximum allowable error was set to zero, CH(U) will not include all of the noninferior extreme points in $\mathbf{N}$, and there may be points outside CH(U) which dominate points on the surface of CH(U). However, we are finding the approximation to $\mathbf{N}$, $\mathbf{N}'$, and a point that is not dominated by any other

point in CH(U) will be considered to be noninferior and a member of $\mathbf{N}'$.

The procedure makes use of the following observations:

Observation 5.1 *By theorem 5.1, if the outward normal,* $\mathbf{w}$*, of a face on the surface of CH(U) is strictly positive, then all points on the face are nondominated with respect to the points in CH(U), and the face forms part of the approximation of the noninferior set.*

Observation 5.2 *By theorem 5.1, if one or more terms of the outward normal,* $\mathbf{w}$*, of a face on the surface of CH(U) are not strictly positive, then the face cannot be noninferior.*

Observation 5.3 *If a noninferior face in* $\mathbf{N}'$ *is contained within a noninferior face of higher dimension, then the face is not included in the definition of* $\mathbf{N}'$*. Thus, the edges of a noninferior face are not included in the definition of* $\mathbf{N}'$*. However, a noninferior edge between two inferior faces must be included in the definition of* $\mathbf{N}'$*. Because the noninferior set of an Linear Problem is connected,* $\mathbf{N}'$ *will never include individual points, and in 3-D problems* $\mathbf{N}'$ *will be made up solely of connected edges and faces. The one exception to this observation occurs if all scenarios share the same scenario-maximal solution, in which case the noninferior set in objective space is the point* $(z_1^{max}, z_2^{max}, z_3^{max})$.

The observations above allow us to determine whether, or not, a face belongs to the approximation. A conclusive test for an edge belonging to the approximation, is also based on Theorem 5.1.

Let: E be an edge between two faces on the surface of CH(U)

F^q be the q-th face incident on edge E, $(q = 1, 2)$

$H(\mathbf{w}^q, d^q)$ be the outward normal of face F^q

The points on edge E are noninferior if and only if there exists a supporting plane at E, $H(\mathbf{w}^e, d^e)$, whose outward normal, $\mathbf{w}^e$, is strictly positive.

A plane supports CH(U) at E if its outward normal can be expressed as a convex combination of the outward normals of the two faces incident on E. Hence E belongs to $\mathbf{N}'$ if and only if the objective function value returned by solving the following problem is strictly positive:

$$\text{maximise} \quad M \tag{5.5 a}$$

$$\begin{aligned} \text{subject to} \quad \alpha^1 w_\omega^1 + \alpha^2 w_\omega^2 &\geq M \quad \text{for } \omega \in \Omega \\ \alpha^1 + \alpha^2 &= 1 \\ \alpha^1,\ \alpha^2 &\geq 0 \end{aligned} \tag{5.5 b}$$

where: w_ω^q is the weight placed on scenario ω by the outward normal of face F^q

M, α^1 and α^2 are the variables

For the objective function value M to be strictly positive, a strictly positive convex combination of the two outward normals must exist.

Observation 5.4 *The objective function value to problem* (5.5) *cannot be strictly positive unless at least one of* (w_1^1 *and* w_1^2) *and one of* (w_2^1 *and* w_2^2) *and one of* (w_3^1 *and* w_3^2) *are strictly positive.*

5.3.6 The XNISE2 Algorithm

Step 0 Initialise $\mathbf{N}' = \phi$
Let F be the set of faces on the surface of CH(U)
Let $E(f)$ be the edges of face $f \in F$ that have been checked
Initialise $E(f) = \phi$ for all $f \in F$

Step 1 Select the next face, f, from F. If all faces have been checked, STOP.

Step 2 If the outward normal of face f, $\mathbf{w}^f$, is strictly positive:

a) add the face to $\mathbf{N}'$.

b) let $E(f)$ = the three edges of face f

c) for each of the three neighbouring faces, $f^n, n = 1, 2, 3$, add the shared edge to $E(f^n)$

d) return to **Step 1**

Step 3 If this step is reached, then the outward normal is not strictly positive, and this face is not noninferior.
If there are 3 edges in $E(f)$ then the edges of this face were checked, when its neighbouring faces were checked, so return to **Step 1**

Step 4 If all terms of the outward normal of this face are negative (i.e. $w_\omega < 0$ for all $\omega \in \Omega$) then the edges of the face cannot be noninferior. (If a neighbouring face is noninferior, the points on the common edge will be noninferior, but they will be included in $\mathbf{N}'$ when that face is included.)

a) let $E(f)$ = the three edges of face f

b) for each of the three neighbouring faces, $f^n, n = 1, 2, 3$, add the shared edge to $E(f^n)$

c) return to **Step 1**

Step 5 If the outward normal of the face has at least one non-negative term, then it may have noninferior edges. For each of its neighbouring faces, $f^n, n = 1, 2, 3$:

a) if the outward normal of the neighbouring face is strictly positive, then the shared edge is contained within that noninferior face, and it should not be included in the definition of $\mathbf{N}'$, so go to c).

b) if, for all three dimensions, at least one of the two faces has a strictly positive term in its outward normal (refer to Observation 5.4), then the edge may be noninferior. So solve problem (5.5).
If the resulting objective function value is strictly positive, the edge is noninferior, so include it in $\mathbf{N}'$.

c) add the shared edge to $E(f)$ and $E(f^n)$

Step 6 When the three neighbouring faces have all been checked, return to **Step 1**

5.3.7 Procedure CHECK

The approximation to the noninferior set consists of a set of faces and edges defined by a set of extreme points. Some of these extreme points may, in fact, be inferior, because they were found using a weighting vector that was not strictly positive. The final step in the process is to identify any inferior extreme points in the approximation, and to replace them with noninferior extreme points. If a point is flagged to show that it was found using a strictly positive weighting vector, then it is known to

be noninferior. If not, problem (5.6) is formed to test whether the point is inferior:

$$\begin{aligned}
\text{maximise} \quad & \Delta = \delta_1 + \delta_2 + \delta_3 && && (5.6\,\text{a})\\
\text{subject to} \quad & z_\omega = (c_0 + c_\omega)\mathbf{x} + \mathbf{q}_\omega \mathbf{y}_\omega && \forall \ \omega \in \Omega && \\
& A\mathbf{x} = \mathbf{b}_0 && && (5.6\,\text{b})\\
& B_\omega \mathbf{y}_\omega + T_\omega \mathbf{x} = \mathbf{b}_\omega && \forall \ \omega \in \Omega && (5.6\,\text{c})\\
& z_\omega - \delta_\omega \geq z_\omega^c && \forall \ \omega \in \Omega && (5.6\,\text{d})\\
& \mathbf{x} \geq 0 && && (5.6\,\text{e})\\
& \delta_\omega \geq 0, \ \mathbf{y}_\omega \geq 0 && \forall \ \omega \in \Omega && (5.6\,\text{f})
\end{aligned}$$

where: δ_ω = the improvement in the objective function value for scenario ω

Δ = the sum of the improvements

z_ω^c = current objective function value for scenario ω

If problem (5.6) has a non-zero objective function value, then the value of at least one of the scenario objective function values has been improved. Because the δ_ω are restricted to be nonnegative, no scenario objective function value can have decreased, and so the original point was inferior.

All criterion vectors that were found to be inferior are replaced in the approximation to the noninferior set by the criterion vectors found by problem (5.6). This is done by calling the subroutine in XNISE1 that adds new points to the set of extreme points, U, and builds the resulting new faces on the surface of CH(U).

After the CH(U) has been updated with all of the new points, procedure XNISE2 is run again to find the new approximation to the noninferior set.

5.4 Presenting the Results

Once an approximation to the noninferior set has been found, the question arises of how to present the results to decision makers. When there are only two scenarios, the noninferior set forms a frontier in two dimensions, and the decision maker can easily see the trade-offs involved in choosing a preferred criterion vector. As discussed in Section 4.7, this frontier is found directly by the solution method. An example of a two-scenario frontier was shown in Figure 4.4.

When there are three scenarios, the noninferior set is a surface in three dimensions, which is rather difficult to represent in two dimensional displays. The available trade-offs are now among three objectives, which means that the decision maker can trade off performance under one scenario against performance under either one, or both, of the other two.

In this section we will consider some simple displays which show the noninferior extreme points only, we will then discuss how these displays can be modified to show the noninferior faces. Use of "Decision Maps" to present the available trade-offs between the scenarios will then be considered. Finally the mapping of the noninferior set onto weighting space is discussed.

5.4.1 Displays of Extreme Points Only

The simplest presentation is to show the noninferior extreme points only. The scenario objective function values for each noninferior extreme point can be presented in tabular form using various sort orders to help the decision maker find patterns and trade-offs. Bar graphs can be used to give pictorial comparisons between the performance of the solutions under the different scenarios, as can value path diagrams. Examples of these forms of presentation are shown in Table 5.4 and in Figures 5.5 and 5.6. The six points, v1 to v6, are the noninferior extreme points of some linear scenario optimisation problem, with three scenarios: 1, 2 and 3.

Noninferior Extreme Points Sorted by Scenario 1			
	Scenario 1	Scenario 2	Scenario 3
v1	41	37	30
v2	45	41	28
v3	48	41	25
v4	48	44	13
v5	50	37	24
v6	58	42	5

Noninferior Extreme Points Sorted by Scenario 2			
	Scenario 1	Scenario 2	Scenario 3
v1	41	37	30
v5	50	37	24
v3	48	41	25
v2	45	41	28
v6	58	42	5
v4	48	44	13

Table 5.4: Noninferior Extreme Points Displayed in Tabular Form

The main shortcoming of these displays is that they only show the noninferior extreme points and provide little information about the noninferior alternatives that lie on the faces and edges of the noninferior set. Although the value paths show the

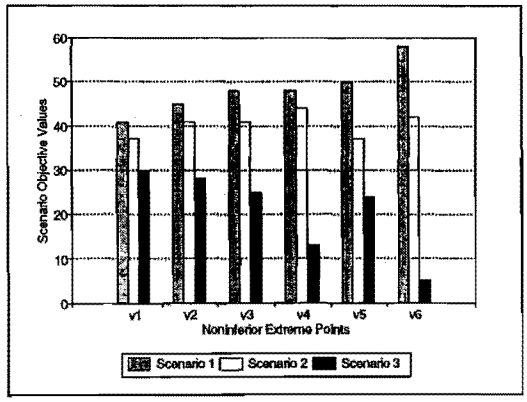

Figure 5.5:
Bar Graph of Scenario Outcomes
The graph shows the scenario objective function values for each noninferior extreme point.

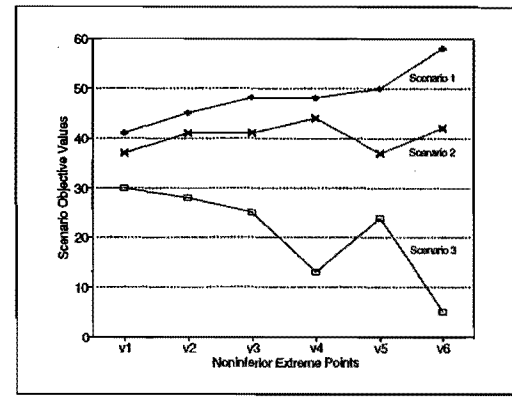

Figure 5.6:
Value Path of Scenario Outcomes
This graph shows the scenario objective function values for each noninferior extreme point, with the points connected by lines to give a picture of the available trade-offs.

extreme points connected by lines, the only purpose of these lines is to make it clear which points belong to which scenarios. This is of particular value when the value paths cross, which will happen for some problems. The adjacencies of the points to one another in the display are determined by the order in which they are arranged along the horizontal axis, and this order can be chosen quite arbitrarily. If two adjacent points on the path are on the same noninferior face, then the line joining them corresponds to a noninferior edge in criterion space. However, if two points that are adjacent on the path are on different faces in the noninferior set, then the line between them will correspond to a line that crosses the interior of the convex hull of the extreme points, and the criterion vectors on that line will be feasible, but dominated. This is because the noninferior extreme points are extreme points of the feasible region of a linear constraint set, and so all points on a line connecting any two of them must be feasible. However, if the two extreme points joined by the line are not vertices of the same face, then the line is not on the surface of the feasible region, but is interior to it, and will be dominated by points on the surface.

5.4.2 Including Representations of the Noninferior Faces

The value path representation can be improved by depicting the noninferior faces on the graph. The noninferior extreme points are joined by lines that correspond to

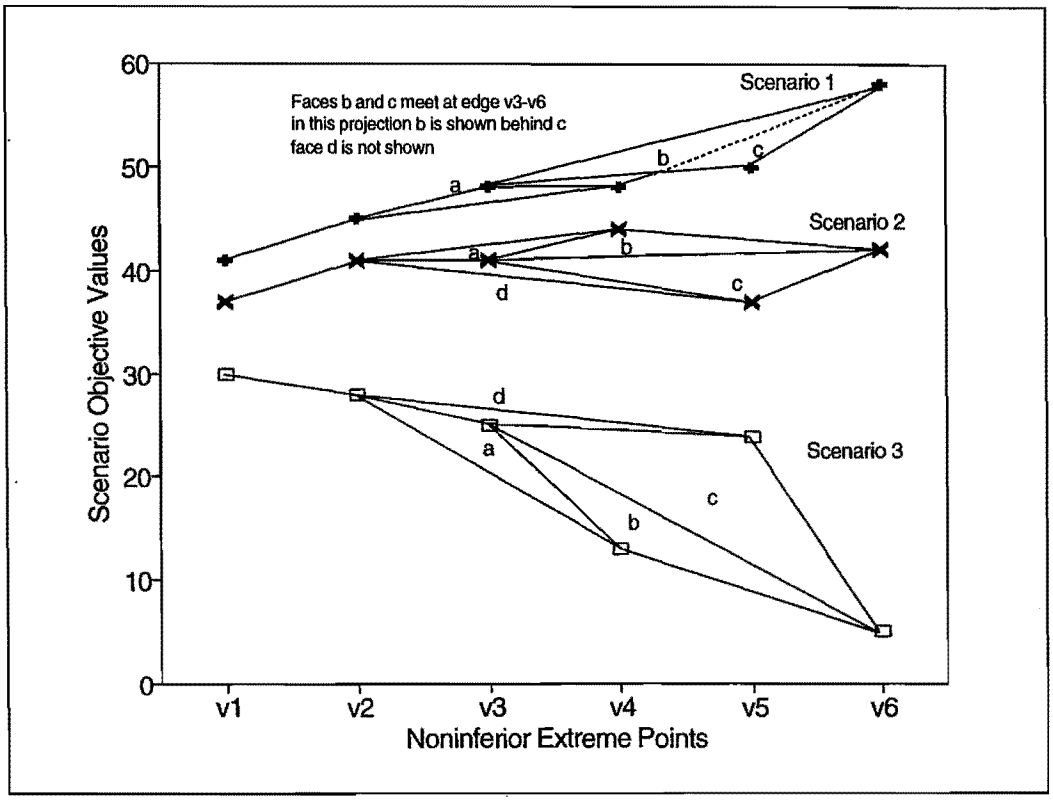

Figure 5.7: Value Path of Scenario Outcomes with Noninferior Faces Shown
This graph shows the scenario objective function values for each noninferior extreme point. The points are connected by lines to represent the faces of the noninferior set.

edges in the noninferior set, so that the noninferior faces appear as regions on the graph of value paths. An example of this is shown in Figure 5.7, which corresponds to the value path shown in Figure 5.6.

The noninferior set, **N**, represented in Figure 5.7 consists of an edge, v1–v2, and four triangular faces, v3–v2–v4, v3–v4–v6, v3–v5–v6 and v3–v2–v5, labelled *a*, *b*, *c*, and *d* respectively. Whereas the value path in Figure 5.6 suggests that the only alternatives to the six extreme points are those points on the lines joining the points, Figure 5.7 shows that there is a much greater range of alternatives than that, being all points on the noninferior faces. The value path in Figure 5.6 also suggests that there are noninferior alternatives along the line v4–v5, but Figure 5.7 shows that this line is not an edge in the noninferior set. It is, in fact, a line that crosses the interior of the convex hull of the noninferior extreme points, and is dominated by faces *b*, and *c*.

When the value path is drawn with the noninferior faces included, the coordinates

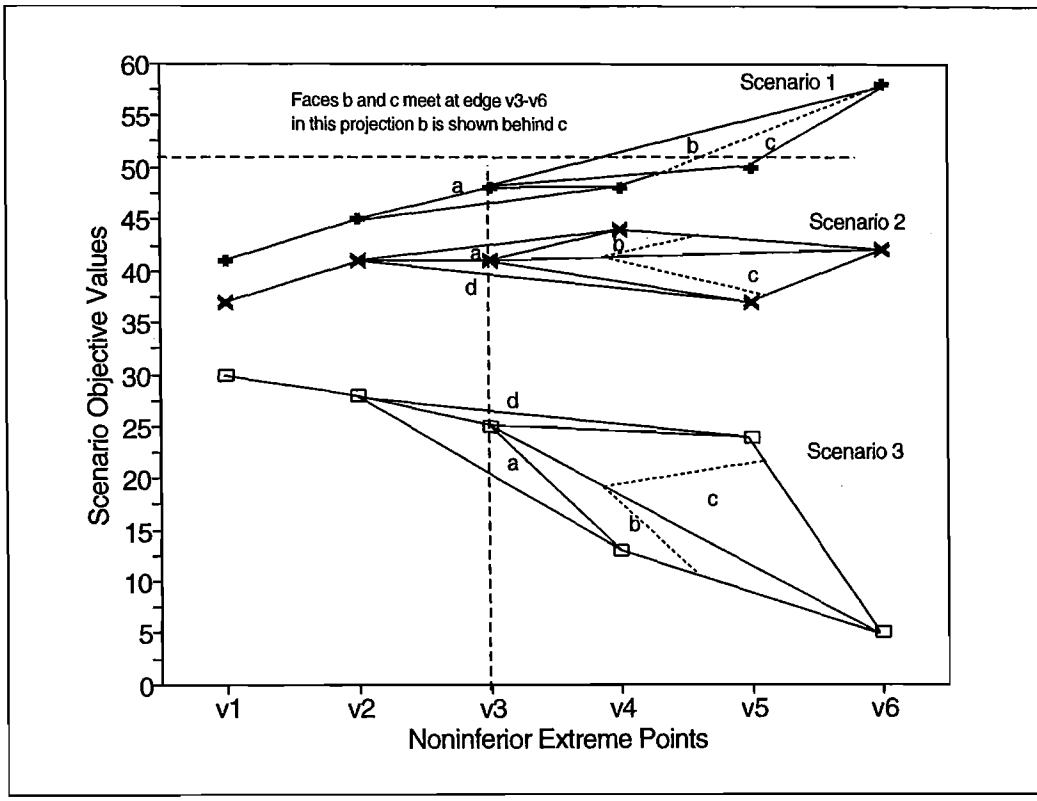

Figure 5.8: Value Path of Scenario Outcomes with Noninferior Faces Shown This graph shows the scenario objective function values for each noninferior extreme point. The fine dotted lines that cross faces b and c illustrate the available trade-offs between scenarios 2 and 3, when the objective function value of scenario 1 is fixed at 51.

of noninferior points can be obtained by drawing vertical lines on the diagram. Consider Figure 5.8, in which the noninferior faces are again shown. A vertical dotted line has been drawn through extreme point v3. It is apparent that the coordinates of v3 lie on the line, and this is the case for all noninferior points that lie on the line. For example, the line crosses the edge v2–v4 three times, once for each scenario, and the coordinates of the corresponding noninferior point can be read off the diagram at those three intersections, giving $z_1 \approx 47$, $z_2 \approx 42.5$, $z_3 \approx 20.5$. Similarly, the point halfway across face a from edge v2–v4 to point v3 can be read off as: $z_1 \approx 47.5$, $z_2 \approx 42$, $z_3 \approx 22.5$.

This provides a graphical method for finding convex combinations of extreme points. In this example, the dotted line through v3 is halfway between extreme points v2 and v4, and so the intersection of the line with edge v2–v4 corresponds to the convex combination $\gamma \mathbf{z}_2 + (1-\gamma)\mathbf{z}_4$ with $\gamma = 0.5$, where $\mathbf{z}_2$, $\mathbf{z}_4$ are the criterion

vectors of points v2 and v4. Similarly, the point inside face a, $\mathbf{z} = (47.5, 42, 22.5)$ is a convex combination of the three extreme points v2, v3 and v4. However, it is important to recognise that valid convex combinations of points can only be found from faces for which the points are all vertices. For example, the intersection of the vertical line through v3 with the edge v2–v5 produces the convex combination $\mathbf{z} = \mathbf{z}_2 * (2/3) + \mathbf{z}_5 * (1/3)$, not a convex combination of points v2 and v4.

For any vertical line on the graph, the relative widths of noninferior regions of each scenario give a visual guide to the sizes of the trade-offs available between the scenarios. For example, it can be seen that the noninferior solutions in the region between v4 and v5 will cover a larger range of objective function values for scenario 3, than for scenarios 1 and 2.

The trade-offs that are available between two scenarios, when the objective function value of the third is held constant, can also be determined from this diagram. As an example, a horizontal line has been drawn at $z_1 = 51$ in Figure 5.8. Vertical lines can be drawn through the points where this contour line intersects the edges of the faces in the noninferior set. These vertical lines identify the intersections in the regions of the other two scenarios. Lines drawn to connect these intersection points show how the objective function values for the other two scenarios can be traded off when $z_1 = 51$. These trade-off lines are shown dotted. To keep the diagram from becoming too cluttered, the vertical lines used to create them are not shown. From the diagram it can be seen that the line $z_1 = 51$ intersects edge v5–v6 at $z_2 \approx 37.5, z_3 \approx 22$. The objective function value of scenario 2 then increases, and that of scenario 3 decreases, as we move across face c to edge v3–v6, and then across face b to edge v4–v6.

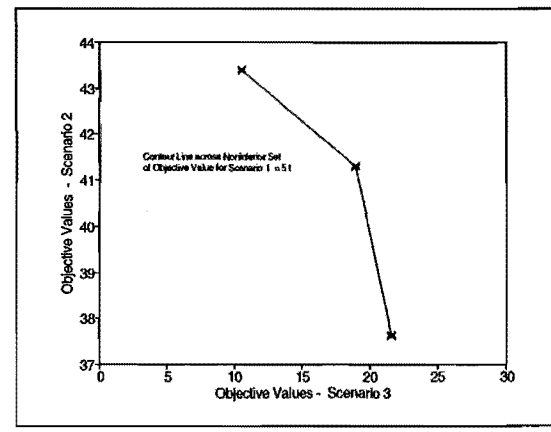

Figure 5.9:
Two-Dimensional Trade-off Frontier
The Two-dimensional trade-off frontier between scenarios 2 and 3, when the objective function value for scenario 1 is held constant at 51.

When the objective function value for one scenario is held constant, a two dimensional trade-off frontier can be drawn for the other two scenarios. For example,

when the objective function value for scenario 1 is held at 51, the trade-offs between scenarios 2 and 3 are as shown in Figure 5.9. This frontier corresponds to the two trade-off lines drawn in Figure 5.8, but is much more readily understandable. A sequence of contour lines drawn on the same diagram produces what is called a "Decision Map", which is discussed next.

5.4.3 Decision Maps

The noninferior set can also be displayed as a projection onto the plane defined by two of the scenario objectives, with the values of the third represented by contour lines. This display is analogous to a geographical map on which contour lines are drawn to show the height. This representation is referred to as a "Decision Map" (Lotov, Chernykh, Bushenkov, Wallenius & Wallenius 1997). Two decision maps for this example are shown in Figures 5.10 and 5.11. In Figure 5.10 the objective function values for scenarios 1 and 2 are shown on the axes, with contour lines used to show the objective function values of scenario 3. The slopes of the contour lines change where they cross from one face to another, and so the intersections of the faces are shown as dotted lines.

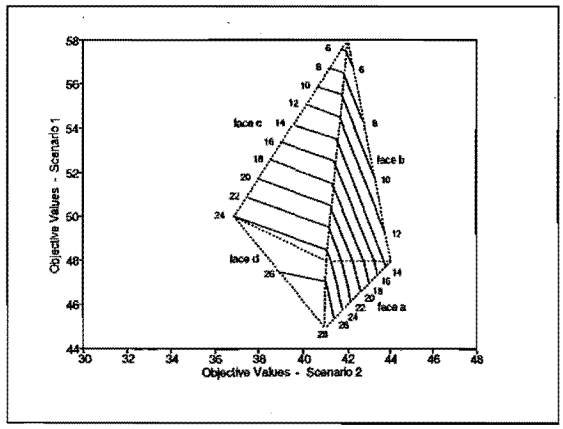

Figure 5.10:
Decision Map with Scenario 3 Shown as Contours
The graph shows the objective function values for scenarios 1 and 2 on the axes, and the objective function values for scenario 3 as contour lines. The edges between the faces are shown as dotted lines.

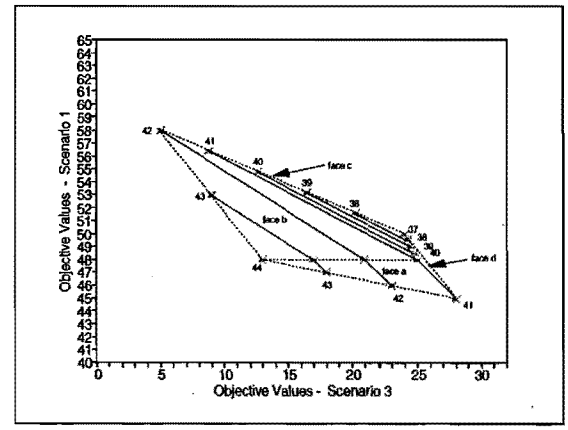

Figure 5.11:
Decision Map with Scenario 2 Shown as Contours
The graph shows the objective function values for scenarios 1 and 3 on the axes, and the objective function values for scenario 2 as contour lines. The edges between the faces are shown as dotted lines.

From Figure 5.10 it can be seen that on face c the slopes of the contour lines are

quite flat, which means that, when the objective function value for scenario 3 is held constant, large changes in the objective function value for scenario 2 are associated with small changes in the objective function value for scenario 1. In Figure 5.11 this can be seen from the fact that the contour lines for scenario 2 are very close together on face *c*. On faces *b* and *a* the contour lines in Figure 5.10 are steeper, indicating that the trade-offs between scenarios 1 and 2 will involve trading large changes in the objective function value for scenario 1 against small changes for scenario 2, when scenario 3 is held constant. In Figure 5.11 this can be seen from the wide spacing of the contour lines for scenario 2 on faces *b* and *a*.

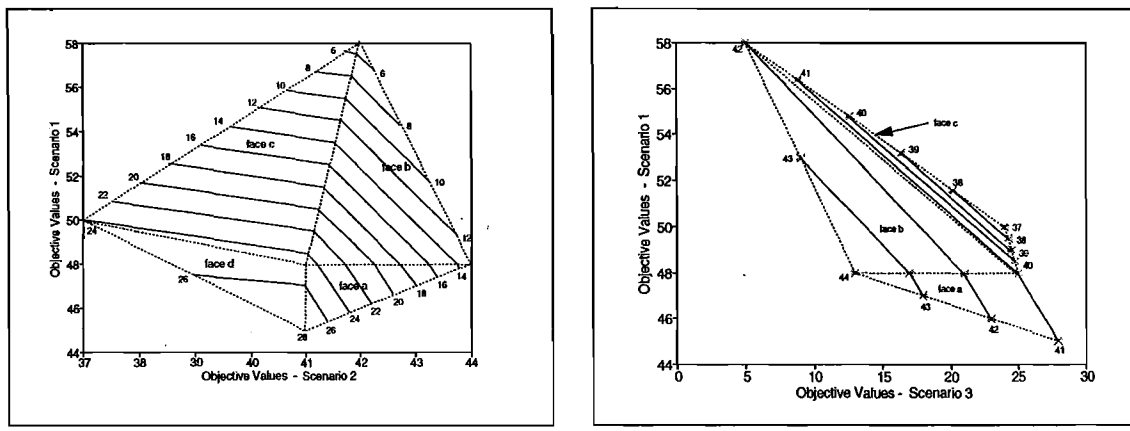

Figure 5.12:
The Decision Maps of Figures 5.10 and 5.11 With Different Scales on the Axes
These graphs use different scales on the axes, with the result that distorted visual impressions are given of the trade-offs between the scenarios.

An aside about decision maps, is that, in common with most graphical presentations, the impressions given by the map are influenced by the scales used on the axes. In Figures 5.10 and 5.11 the axes have been scaled so that a unit change in objective function value corresponds to the same distance on either axis. In Figure 5.12 the axes have been chosen to so that the diagrams fill the available space. This makes it easier to see the detail, but gives a distorted impression of the available trade-offs, both in terms of size and slope.

5.4.4 The Noninferior Set in Weighting Space

The noninferior set can also be mapped into weighting space, in which there is a dimension for each scenario. The scenario weights over the noninferior set are

nonnegative, and they lie on a plane that is inclined in all three dimensions. When the weights are normalised to sum to 1, all of the weight vectors lie on the triangular plane:

$$w_1 + w_2 + w_3 = 1 \qquad \text{where } 0 \leq w_i \leq 1 \quad \text{for } i = 1, 2, 3$$

It is convenient to project this plane onto two dimensions, as in Figure 5.13, which shows the weighting space for this example. The triangular plane can be visualised as sloping down from the point 1 unit above the origin (that is $\mathbf{w} = (0, 1, 0)$) to the diagonal line $w_2 = 0$.

Each region in weighting space corresponds to an extreme point in criterion space, and that point is the preferred solution for the range of weights within the region. For example, point v1 is the preferred solution for all weighting vectors such that $w_3 \geq 2/3$. Each line in weighting space separates the regions of two points, and corresponds to the edge joining those points in criterion space. For example, the line separating regions v1 and v2 in Figure 5.13 corresponds to the edge v1–v2 in Figure 5.7, and the line separating regions v5 and v3 corresponds to the edge v5–v3. The intersection points of lines in weighting space correspond to faces in criterion space. In Figure 5.13 face a is at the intersection of the three lines that correspond to its three edges. That is, edges v3–v4, v4–v2, v2–v3.

These observations reflect the fact that all points on a noninferior face are preferred decisions for a single weighting vector, whereas noninferior extreme points and edges are preferred decisions for ranges of scenario weights. An edge in criterion space appears as a line in weighting space, one end of which is at the coordinates of the weighting vector of the face on one side of it, and the other end is at the weighting vector of the face on its other side. For an edge between two noninferior faces, all terms of the weighting vector at each end are strictly positive, because the outward normal of a noninferior face is strictly positive. An edge between a noninferior face and an inferior face will have one end on the perimeter of the triangle of valid weights, where one of the scenarios has a weight of zero. An isolated noninferior edge, such as v1–v2 has an inferior face on both sides, and so both of its ends are on the perimeter of the weighting triangle. The weights where an edge meets the perimeter of the weighting triangle will generally not be the weights of the inferior face on that side of the edge, because the zero term will actually be negative in the weighting vector of the inferior face.

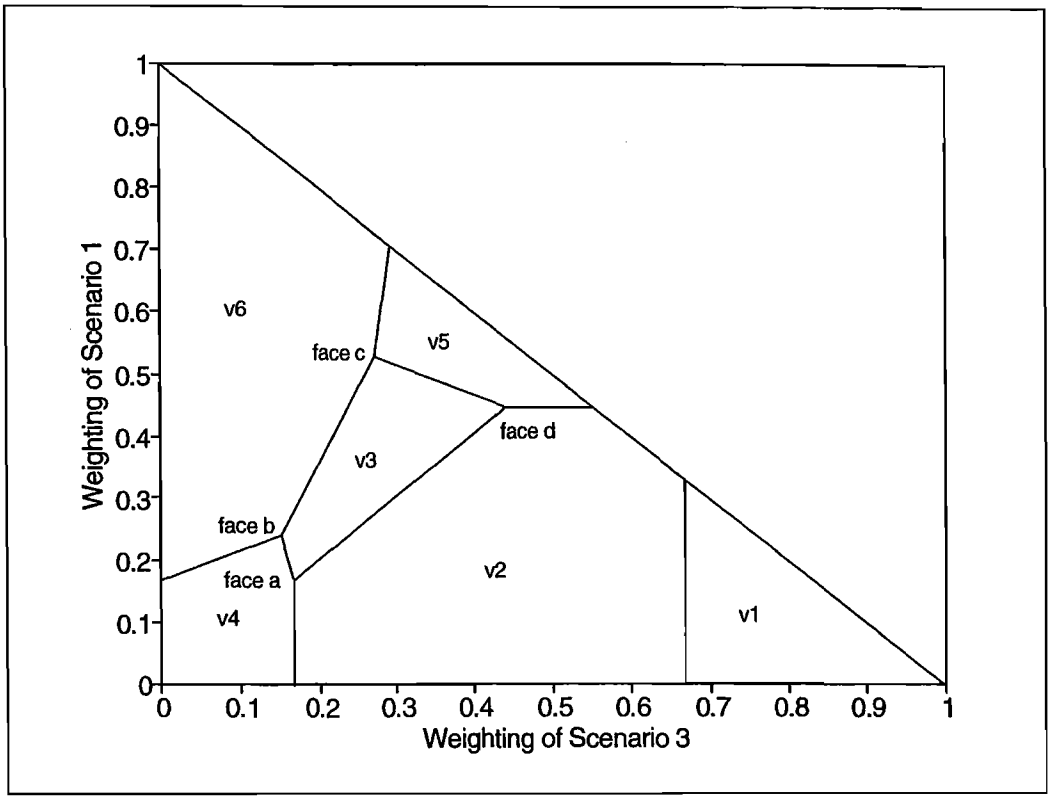

Figure 5.13: The Noninferior Set Mapped Onto Weighting Space

The scenario weights can be interpreted as being the scenario probabilities of the stochastic optimisation formulation of the problem, and the representation in weighting space can be used to find the optimal solution for any setting of the probabilities. For example, solution v2 is optimal for the probability vector $\mathbf{p} = (0.2, 0.4, 0.4)$. If the original problem is to be solved as a stochastic optimisation problem maximising the expected value over the scenarios, then it is apparent from the representation in weighting space that the noninferior extreme points are much more robust with respect to the scenario probabilities than are the noninferior faces. This is because the faces are optimal for a single probability vector, whereas the extreme points are optimal for a range of probability vectors.

5.4.5 Interactively Reducing the Noninferior Set

On considering the noninferior set, the decision maker may decide that some regions of it are clearly unacceptable. For example, on viewing Figure 5.7, the decision maker may decide that 13 is the minimum acceptable objective function value for any of

the scenarios. This decision implies the addition to the problem of the constraint $z_\omega \geq 13 \; \forall \, \omega \in \Omega$ which renders a section of the current noninferior set infeasible. The representations of the noninferior set can then be redrawn without this region, as is illustrated in Figure 5.14. The new corner points, labelled v7 and v8 in Figure 5.14, can be found from the geometry of the noninferior set in the same manner as was discussed in Section 5.4.2. On Figure 5.7 vertical lines can be drawn through the points where the line $z = 13$ intersects with the faces b and c. The intersections of these vertical lines with faces b and c give the coordinates of points v7 and v8. When these two points have been found, the pruned version of the noninferior set can be drawn.

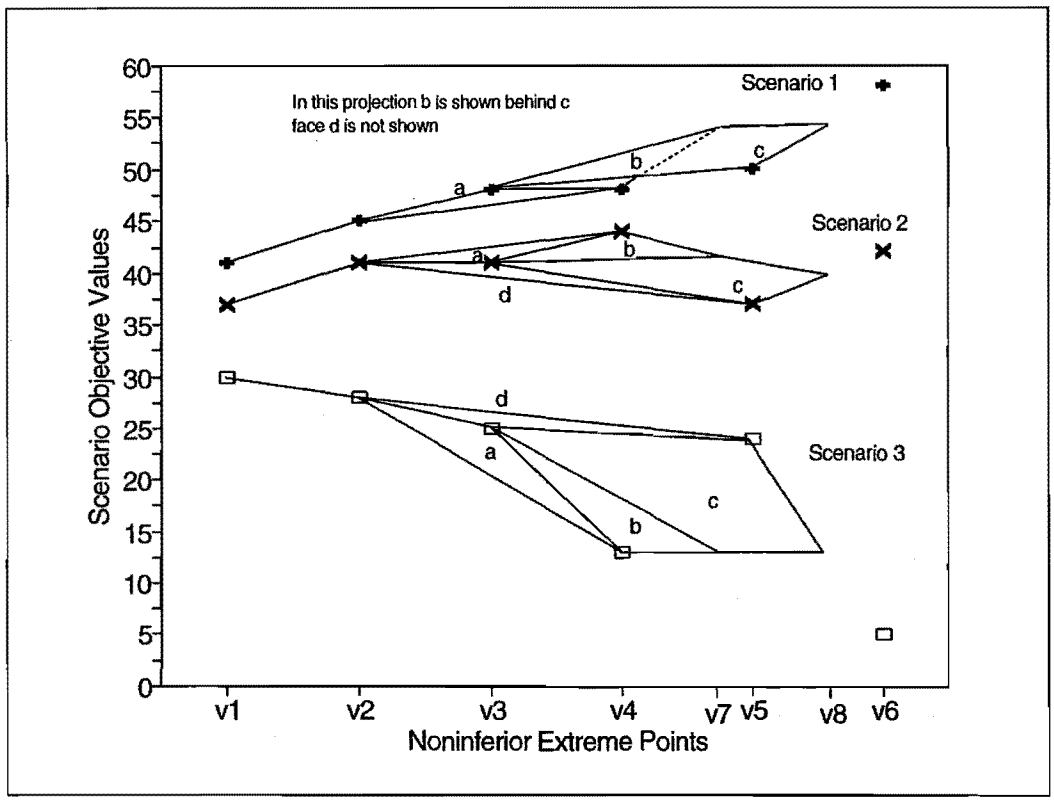

Figure 5.14: The Noninferior Set Pruned at $z_\omega \geq 13 \; \forall \, \omega \in \Omega$
The noninferior set after setting the minimum acceptable objective function value to 13 for all scenarios.

The decision maker may wish to prune more regions from the noninferior set to gradually narrow down the range of choices.

5.4.6 Summary

We have discussed several different ways of presenting information about the noninferior set, each of which leads to different insights about the problem, and each of which may be used for different purposes. The tabular and graphical representations of the noninferior extreme points, as in Table 5.4 and Figures 5.5 and 5.6, describe the extreme points in the noninferior set, but provide no information about the noninferior faces. If the decision maker wants to find the optimal, expected value solution to the stochastic optimisation problem then it is the extreme points that he or she will be interested in, along with the information available from the representation in weighting space about which solutions are optimal for various scenario probabilities.

On the other hand, some decision makers will want to explicitly trade off between the scenarios when choosing a solution to implement, and they will want information about the whole noninferior set, not just the extreme points. This information is provided by the including the noninferior faces on the value path diagram, as in Figure 5.7, and by producing decision maps, such as Figures 5.10 and 5.11. These representations enable decision makers to visualise the trade-offs involved in choosing a solution to implement. The decision makers can then choose a final criterion vector that matches their risk preferences, whether they be risk takers, risk neutral, or risk averse.

The representation in weighting space can be used to inform the decision makers about the relative weights that they are implicitly assigning to the scenarios by their choice of criterion vector. These weights can be thought of weights reflecting the relative importance of the scenarios, or as probabilities, in which case the representation in weighting space shows the scenario probabilities for which their choice of criterion vector is the optimal solution to the expected value problem.

5.5 Selection of Non-extreme Criterion Vectors

When looking for a solution to implement, the decision maker works in criterion space, and selects a criterion vector that best matches his or her preferences. However, selection of a criterion vector does not complete the process; the decision maker needs a solution to implement. This means that the selected criterion vector must

be translated back into solution space to obtain a solution vector. If the selected criterion vector is an extreme point, then a corresponding solution vector is already known from the analysis. However, if the selected criterion vector is not an extreme point, then the corresponding decision vector is not already known. This means that the problem must be solved again, or a decision vector found by using the relationships between the geometry of the noninferior set in objective space, and the geometry of the efficient set in solution space. For large problems it will be quicker to use the geometry of these spaces to find the solution vector, rather than solving the problem again. However, although there is a one-to-one mapping from solution space to criterion space, the mapping from criterion space to solution space may be one-to-many. This means that the decision maker's choice of a preferred criterion vector may not identify a unique solution for implementation. Three propositions about the relationships between the noninferior and efficient sets are given next, before these issues are discussed further.

When the XNISE algorithm has been run for problem (5.2), on page 116, we will have an approximation to the noninferior set, $\mathbf{N}'$, characterised by a set of extreme points that are known to be noninferior. A non-extreme criterion vector, $\hat{\mathbf{z}} \in \mathbf{N}'$, may be expressed as the convex combination of two, or more, extreme points:

$$\hat{\mathbf{z}} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)\gamma$$

where: γ is the convex combination vector, such that

$$\sum_{t=1}^{T} \gamma_t = 1 \quad \text{and} \quad 0 \leq \gamma_t \leq 1 \quad \text{for} \quad t = 1, 2, \ldots T$$

However, some of these extreme points may correspond to multiple solution vectors, only one of which will have been found by the solution process. Let $\mathbf{x}_t^{i_t}$ be the set of extreme points in solution space that evaluate to the criterion vector $\mathbf{z}_t$, where: $i_t = 1, 2, \ldots, I_t$; indexes those extreme solutions.

That is:

$$C\mathbf{x}_t^{i_t} = \mathbf{z}_t \qquad \text{for} \qquad i_t = 1, 2, \ldots, I_t$$

where the rows of C are the coefficients of the scenario objective functions

Proposition 5.1 *If* $\mathbf{z}_1, \ldots, \mathbf{z}_T$ *are vertices of the same face in the approximation,* $\mathbf{N}'$, *then* $\hat{\mathbf{z}} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)\gamma$ *is in* $\mathbf{N}'$.

Proof: This follows trivially from the definition of a convex combination.

Proposition 5.2 *If* $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$ *are vertices of the same face in the approximation,* $\mathbf{N}'$*, and* $\hat{\mathbf{z}} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)\gamma$*, then the solution vector* $\hat{\mathbf{x}} = (\mathbf{x}_1^{i_1}, \mathbf{x}_2^{i_2}, \ldots, \mathbf{x}_T^{i_T})\gamma$ *evaluates to* $\hat{\mathbf{z}}$*, and is a member of the approximation of the efficient set, where, for each* $\mathbf{z}_t$*,* $\mathbf{x}_t^{i_t}$ *is one of the extreme points in solution space such that* $C\mathbf{x}_t^{i_t} = \mathbf{z}_t$*.*

Proof:

$$\begin{aligned} C\hat{\mathbf{x}} &= (C\mathbf{x}_1^{i_1}, C\mathbf{x}_2^{i_2}, \ldots, C\mathbf{x}_K^{i_K})\gamma \\ &= (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K)\gamma \\ &= \hat{\mathbf{z}} \end{aligned}$$

So $\hat{\mathbf{x}}$ evaluates to $\hat{\mathbf{z}}$ as required, and because $\hat{\mathbf{z}} \in \mathbf{N}'$ $\hat{\mathbf{x}}$ is, by definition, a member of the approximation to the efficient set.

When $I_t > 1$ for at least one of t, then there will be an infinite number of solution vectors that evaluate to $\hat{\mathbf{z}}$, and the one that is found when translating from criterion space to solution space will depend on which of the $\mathbf{x}_t^{i_t}$ solution vectors were found by XNISE.

Proposition 5.3 *Let* $\mathbf{z}_1, \ldots, \mathbf{z}_T$ *be vertices of the same face,* f*, in the approximation,* $f \in \mathbf{N}'$*. If* $f \in \mathbf{N}$ *i.e. face* f *is a member of the true noninferior set, then* $\hat{\mathbf{x}}$ *is an efficient solution. If face* f *is not a member of the true noninferior set,* $f \notin \mathbf{N}$*, then* $\hat{\mathbf{x}}$ *is a dominated, feasible solution.*

Proof:

i) If the face is noninferior, $f \in \mathbf{N}$, then $\hat{\mathbf{z}}$ is noninferior, and so, by definition, $\hat{\mathbf{x}}$ is efficient.

ii) If the face is inferior, $f \notin \mathbf{N}$, then a feasible criterion vector, $\bar{\mathbf{z}}$, $\bar{\mathbf{z}} \neq \hat{\mathbf{z}}$, exists such that $\bar{\mathbf{z}} \geq \hat{\mathbf{z}}$, and so $\hat{\mathbf{x}}$ is dominated. However, all criterion vectors in the approximation $\mathbf{N}'$ are feasible, which means that $\hat{\mathbf{z}}$ is feasible, and thus, so is $\hat{\mathbf{x}}$.

Once the decision maker has chosen a criterion vector one of six possible situations will apply:

1. The criterion vector is an extreme point and has a unique solution vector.

2. The criterion vector is an extreme point and has multiple solution vectors.

3. The criterion vector is in a noninferior face, the vertices of which have unique solution vectors.

4. The criterion vector is in a noninferior face, the vertices of which have multiple solution vectors.

5. The criterion vector is in a dominated face, the vertices of which have unique solution vectors.

6. The criterion vector is in a dominated face, the vertices of which have multiple solution vectors.

Situations 1 and 3 are equivalent, with situation 1 being a special case of situation 3. The chosen criterion vector, $\hat{\mathbf{z}}$, will be noninferior, and its convex combination operator, γ, can be used to find the corresponding, efficient, and unique, solution vector. That is, $\hat{\mathbf{x}} = (\mathbf{x}_1^{i_1}, \mathbf{x}_2^{i_2}, \ldots, \mathbf{x}_T^{i_T})\gamma$ where γ is the convex combination vector in $\hat{\mathbf{z}} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)\gamma$.

Situations 2 and 4 are equivalent, with situation 2 being a special case of situation 4. Use of the convex combination operator will produce an efficient solution vector, but one that is not unique. If the decision maker is only concerned with achieving the preferred criterion vector, and does not wish to choose which efficient solution vector is actually implemented, then the analysis is complete. However, if the decision maker wants to choose between the solutions that evaluate to the preferred criterion vector, then additional work is required.

One approach is to find the alternative optimal solutions for presentation to the decision maker. The original problem, problem (5.2) is reproduced here for convenience as problem (5.7).

$$\begin{aligned}
&\text{maximise} && V = w_1 z_1 + w_2 z_2 + w_3 z_3 && && (5.7\,\text{a})\\
&\text{subject to} && z_\omega = (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x} + \mathbf{q}_\omega \mathbf{y}_\omega && \forall\ \omega \in \Omega && (5.7\,\text{b})\\
& && A\mathbf{x} = \mathbf{b}_0 && && (5.7\,\text{c})\\
& && B_\omega \mathbf{y}_\omega + T_\omega \mathbf{x} = \mathbf{b}_\omega && \forall\ \omega \in \Omega && (5.7\,\text{d})\\
& && z_\omega \geq z_\omega^{lb} && \forall\ \omega \in \Omega && (5.2\,\text{e})\\
& && \mathbf{y}_\omega \geq 0,\ \mathbf{x} \geq 0 && && (5.7\,\text{f})
\end{aligned}$$

where: $\mathbf{w}$ = the outward normal of the face that contains the preferred criterion vector. In situation 2, the outward normal of any of the faces for which the preferred criterion vector is a vertex can be used

z_ω^{lb} = the coordinates of the preferred criterion vector

Once the alternative optima have been found, the decision maker can choose among them, or investigate using a convex combination of them.

Another approach is to solve problem (5.7) with a new objective function to discriminate between the alternative optima. A suitable approach would be find the stage one decision that performs best for stage one, while still producing the preferred vector of scenario outcomes. That is, to maximise for the objective $\mathbf{c}_0\mathbf{x}$, while satisfying the constraint set (5.7 b) to (5.7 f).

In situation 5, use of the convex combination operator will produce a unique solution vector, but one that is dominated. The decision maker may choose to accept this solution as being good enough, but an efficient solution can be found by solving problem (5.7). It would seem appropriate to use the weighting vector of the face that contains the selected criterion vector to form the objective function, because these weights will maintain the trade-offs between the scenarios that were implied by the choice of criterion vector. However, any strictly positive weighting vector could be used. Another approach would be to use preemptive goal programming so that the objective function value for one of the scenarios is improved as much as possible before the others are improved. Whichever approach is used, if the approximation to the noninferior set was found with a small maximum permitted error value, then the change in the criterion vector can be expected to be small.

The other way of handling situation 5 would be to run the XNISE algorithm again to find the portion of the exact noninferior set that dominates the chosen face. It is not possible to work on the chosen face alone because a neighbouring face may also be dominated, and the addition of a new extreme point may mean that the neighbouring face should also be replaced, as was illustrated in Figure 5.4 on page 129. However, to prevent XNISE selecting any other faces, their maximum possible errors can be set to zero as part of the initialisation. This will prevent them being selected for improvement, but will allow XNISE to replace some of them if necessary. To find the desired portion of the noninferior set exactly, the maximum allowable error would be set to zero.

If a small maximum allowable error was used to find the original approximation, then improving the approximation may not achieve much. However if a large maximum allowable error was used, the increase in information may be significant. When the required portion of the exact noninferior set has been found, the decision maker must select a new preferred criterion vector. One of situations 1, 2, 3 and 4 will then apply.

Situation 6 is similar to situation 5, in that a dominated solution vector will be found, but this solution vector is not unique. There will be other solutions that produce the same criterion vector. As for situation 5, problem (5.7) can be solved to find an efficient solution that at least matches the scenario objective function values of the chosen criterion vector. This solution could then be checked to see if there are multiple optima and, it there are, one of the approaches discussed for situation 4 can be used.

Otherwise, as for situation 5, the XNISE algorithm can be run again to find the portion of the exact noninferior set that dominates the chosen face, and the decision maker can then choose a new preferred criterion vector. One of situations 1, 2, 3 and 4 will then apply.

5.5.1 Summary

Selection of a preferred criterion vector by the decision maker must be followed by the translation of that vector into a solution that can be implemented. When the preferred criterion vector really is noninferior, and the noninferior extreme points that characterise the face that contains this vector have unique solutions, then this translation is straightforward, and will produce a unique, efficient solution vector. However, when the preferred criterion vector is actually an inferior vector, its corresponding solution vector will be dominated, and the decision maker may wish to find a solution vector that is efficient. Furthermore, if some of the noninferior extreme points that characterise the face that contains the preferred vector have multiple solutions, then there will be an infinite number of solution vectors that evaluate to the preferred criterion vector, and the decision maker may wish to choose among them according to additional criteria. In this section we have discussed various combinations of these outcomes, and we have considered methods of dealing with them to bring the search for a final solution to a conclusion.

5.6 Conclusions

In Chapter 5 we have taken the basic ideas developed in Chapter 4 for problems with two scenarios, and extended them to deal with the greater complexities presented by problems with three scenarios. A set of algorithms has been developed to find an approximation to the noninferior set in criterion space. This set provides the decision maker with alternative stage one decisions to choose among, and with insights about the trade-offs that must be made between the scenarios when choosing a decision to implement.

Characterisation of an approximation to the noninferior set in criterion space is a necessary first step in dealing with the problem, but decision makers need a means of translating from a preferred criterion vector to a decision that can be implemented. These issues were addressed in the last two sections. In Section 5.4 methods of representing the noninferior set were discussed. These include simple displays that report the noninferior extreme points only, and provide no information about the rest of the noninferior set. One of these displays, the value path graph, can be extended to show the faces of the noninferior set. This graph can be used to consider trade-offs and to find the coordinates of non-extreme criterion vectors. Another approach is to present the noninferior set as a two-dimensional decision map, in which the objective function values of two scenarios are shown on the axes, and the values of the third are represented as contour lines. The different displays provide different insights into the problem situation and the available solutions, and it is likely that decision makers will find it useful to have all of them available to work with. These displays can be manipulated and refined interactively with graphical software such as Matlab. Generally available spreadsheet packages can be used to draw the graphs and do simple manipulations, such as changing the axes of graphs and changing sort orders.

The final section discusses translation from a selected vector in criterion space to a suitable solution vector in solution space. For at least some problems this will not be straightforward because there can be many extreme points in solution space that map to the same noninferior extreme point in criterion space. This means that selection of a criterion vector cannot be relied upon to uniquely identify a single solution vector, and a method is required to find the solution vector that best matches the decision maker's preferences. This may be done by introducing

additional criteria to drive the selection of the solution vector. One possibility is to maximise the stage one return (or minimise the stage one investment) while still achieving the decision maker's preferred vector of scenario objective function values.

Finally, the decision maker will have chosen a criterion vector from an approximation to the noninferior set, and the selected criterion vector may, in fact, be inferior, and its solution vectors dominated. The decision maker will have declared his or her preference structure by his or her choice of criterion vector. This preference structure can be assumed to be expressed by the scenario weights of the equation of the noninferior face. An efficient solution can then be found by solving the problem again using these weights, and using the preferred criterion vector as lower bounds on the scenario objective function values. If the final criterion vector has alternative optima, the decision maker may use additional criteria to choose among them.

Noninferior Set Scenario Analysis has been developed as an approach to scenario optimisation that provides the decision maker with a set of non-dominated solutions to choose among, rather than producing a single solution that is optimal for a particular choice of scenario probabilities. This choice of alternatives to choose among means that the decision maker can consider non-quantifiable issues when selecting a final decision. It also means that the decision maker is not required to assign probabilities to the scenarios in order to obtain a solution, and no assumptions need to be made about the decision maker's attitude to risk.

The algorithms used in this chapter are based on the work of Solanki et al. (1993), although we have simplified them by taking advantage of working in only three dimensions. We have implemented these algorithms using "AMPL" (*A Modelling Language for Mathematical Programming*) and solved the examples presented in Chapter 6 using this implementation. The contribution of this work to the literature is the conceptualisation of the scenario objectives as competing objectives, and the resulting formulation of the problem as a multiobjective problem. We have used the XNISE algorithm to solve the resulting problem, but we have not extended the algorithm itself.

Chapter 6

Example Problems

6.1 Introduction

In this chapter we use Noninferior Set Scenario Analysis to analyse two small example problems from the literature. Both examples are two stage capacity expansion problems from the electricity sector. In the first example, by Louveaux and Smeers (Louveaux & Smeers 1988), an uncertain future demand is to be met by building a mix of four types of generating plant that have various construction and operating costs. The uncertain future demand is represented by three scenarios. The objective is to choose the plant configuration that minimises the cost of meeting demand. Because demand must be met under all scenarios the scenario with the greatest demand determines the total capacity that must be built, and the decision maker's choices are limited to deciding the mix of plant. Because the cost turns out to be insensitive to the mix of plant, the problem is rather uninteresting, and we modify the example to make it a profit maximisation problem in which the decision maker can leave demand unmet under some, or all, of the scenarios.

In the second example, by Infanger (Infanger 1992, Infanger 1994), two types of generator can be built to meet an uncertain demand for electricity. The two types of generator have uncertain availabilities, and different construction and operating costs. Electricity can be bought from outside to meet demand, but at a relatively high cost. The availabilities that will actually be achieved by each of the two types of plant are uncertain, and are represented by discrete random variables. The load duration curve of the electricity demand is approximated by three steps. The duration of each step is known, but the demand at each step is uncertain. These

uncertain demands are also represented by discrete random variables.

The uncertainties are modelled as scenarios, one for every possible combination of the values of the random variables, which, in this example, produces 1280 scenarios. We use NSSA with just three summary scenarios to find a solution. The surface of the objective function of this problem turns out to be rather flat, and NSSA is able to find good solutions with considerably less effort than is required to solve the full stochastic formulation. More importantly, NSSA produces many more insights about the problem than are available from the single solution produced by stochastic optimisation, including the insight that the solution surface is flat.

6.2 Louveaux and Smeers Test Problem

6.2.1 The Original Problem

Louveaux & Smeers (1988) used this problem to illustrate the application of stochastic optimisation to a capacity expansion problem. It is a two-stage linear optimisation problem in which a decision maker must install generating capacity to meet an uncertain future demand for electricity. Currently the decision maker has no generating capacity. There are four types of plant available, all of which have a construction lead time equal to stage one, but which vary in their construction and operating costs. There are limited funds available with which to build the generating capacity. The demand is represented as three modes: base load which has a load duration of 100%, middle with a load duration of 60%, and peak with a load duration of 10%. The middle and peak loads are known with certainty, but demand for base load is uncertain, and this uncertainty is summarised as three scenarios with known probabilities. The data for the problem is summarised in Table 6.1.

	Scenario				Plant Type			
	High	Average	Low		P1	P2	P3	P4
Base Load	7	5	3	Construction Cost	10	7	16	6
Middle Load	3	3	3	Operating Cost	40	45	32	55
Peak Load	2	2	2					
Probability	0.3	0.4	0.3	Available Funds: 120				

Table 6.1: Data for the Louveaux and Smeers Example
The construction and operating costs for the plant types are for one unit of capacity. One unit of plant can meet one unit of demand for any mode.

The example is formulated as problem (6.1):

$$\text{minimise} \quad Z = \sum_{\omega} p_\omega z_\omega \tag{6.1 a}$$

$$\text{subject to} \quad z_\omega = \sum_{g=1}^{4} bc_g x_g + \sum_{g=1}^{4}\sum_{m=1}^{3} pc_{gm} ld_m y_{gm\omega} \qquad \forall\ \omega \in \Omega$$

$$\sum_{g=1}^{4} bc_g x_g \leq F \tag{6.1 b}$$

$$\sum_{m=1}^{3} y_{gm\omega} \leq x_g \quad \text{for } g = 1,\ldots,4 \quad \forall\ \omega \in \Omega$$

$$\sum_{g=1}^{4} y_{gm\omega} \geq D_{m\omega} \quad \text{for } m = 1,\ldots,3 \quad \forall\ \omega \in \Omega$$

$$\mathbf{y} \geq 0,\ \mathbf{x} \geq 0$$

where: Ω = set of scenarios

p_ω = probability of scenario ω

z_ω = cost under scenario ω

bc_g = building cost for plant type g

pc_{gm} = production cost for plant type g supplying demand mode m

ld_m = load duration of mode m

x_g = size of plant g to build

$y_{gm\omega}$ = despatch of plant g to demand mode m under scenario ω

F = funds available for building new generating capacity

$D_{m\omega}$ = demand for mode m under scenario ω

Because the decision maker is required to meet demand under all scenarios, at least 12 units must be installed at stage one.

The stage one decision is the vector of plant capacities to be installed, and the recourse decisions are the despatches of the plant types to the demand modes in stage two. The optimal expected value solution is shown in Table 6.2. In this solution the funds constraint is binding.

Because this example is very tightly constrained by the limitation on initial investment and the requirement that demand be met under all scenarios, each scenario-maximal solution is almost optimal for both of the other scenarios. This

Expected Profit: 381.85										
Scenario (profit):		High (470.33)			Average (380.33)			Low (295.4)		
		Despatch			Despatch			Despatch		
Plant	Built	Base	Middle	Peak	Base	Middle	Peak	Base	Middle	Peak
P1	2.67	2.67	0	0	1.67	1	0	0	2.67	0
P2	4	1	3	0	0	2	2	0	0	2
P3	3.33	3.33	0	0	3.33	0	0	3	0.33	0
P4	2	0	0	2	0	0	0	0	0	0

Table 6.2: Optimal Expected Value Solution to the Original Problem
The 'Built' column shows the capacity to be built for each plant type. The remaining columns show the despatch to each mode under each scenario. For example, this solution builds 4 units of plant P2. Under the High scenario, 1 unit is despatched to meet base load, and 3 units are despatched to meet middle load. Under the Average scenario, 2 units are despatched to each of middle and peak loads, while, under the Low scenario, 2 units are despatched to meet peak load and 2 units are left unused.

Scenario	Plant Built				Objective Values			Funds
Optimised	P1	P2	P3	P4	High	Average	Low	Used
High	4.17	3	2.83	2	469.33	381.33	297.83	120
Average	0.83	3	4.17	4	480.67	378.67	294.40	120
Low	3	2	3	4	481.00	381.00	293.00	116
	Percentage Differences:				2.5%	0.7%	1.7%	

Table 6.3: Scenario-Maximal Decisions With Limited Funds
The capacity expansion decisions are constrained by the need to meet demand under all scenarios, and by the available funds being limited to 120.

can be seen from Table 6.3 which lists the three scenario-maximal solutions and their performances under each scenario. The greatest percentage regret suffered when implementing a scenario-maximal solution and having another scenario occur is 2.5%. When compared with the imprecision inherent in gathering data about the future, this is not significantly different from zero. Table 6.4 lists the three scenario-maximal solutions and their performances under all scenarios when the available funds are increased sufficiently to make the funds constraint slack under all scenarios. The increase in available funds allows the choice of plant types to be tailored to match each scenario, and so the variation in performance between the three solutions is increased. However, the variation is still very small. In this example there is little to be gained from including uncertainty in the formulation, and so we have modified it to make it more interesting.

Scenario	Plant Size				Objective Values			Funds
Optimised	P1	P2	P3	P4	High	Average	Low	Used
High	0	3	7	2	461.00	379.40	306.30	145
Average	1	4	5	2	467.00	377.00	297.40	130
Low	3	2	3	4	481.00	381.00	293.00	116
	Percentage Differences:				4.3%	1.1%	4.5%	

Table 6.4: Scenario-Maximal Decisions With Unlimited Funds
The capacity expansion decisions are constrained by the need to meet demand under all scenarios, but the available funds are sufficient to build the best configuration for each scenario.

6.2.2 A Modified Version

We have dispensed with the requirement that demand be met under all scenarios, and changed the objective to be profit maximisation. Because demand no longer has to be met the decision maker is now free to supply as much, or as little, of the demand as he wishes, and it is allowable to install capacity that is insufficient to meet all of the demand under some, (or even all) of the scenarios. The modified example is problem (6.2):

$$\text{maximise} \quad Z = \sum_{\omega} p_\omega z_\omega \tag{6.2 a}$$

$$\text{subject to} \quad z_\omega = -\sum_{g=1}^{4} bc_g x_g + \sum_{g=1}^{4}\sum_{m=1}^{3} (P_m - pc_{gm}) ld_m y_{gm\omega} \quad \forall\ \omega \in \Omega$$

$$\sum_{g=1}^{4} bc_g x_g = f \tag{6.2 b}$$

$$\sum_{m=1}^{3} y_{gm\omega} \le x_g \qquad \text{for } g = 1, \ldots, 4 \qquad \forall\ \omega \in \Omega$$

$$\sum_{g=1}^{4} y_{gm\omega} \le D_{m\omega} \qquad \text{for } m = 1, \ldots, 3 \qquad \forall\ \omega \in \Omega$$

$$\mathbf{y} \ge 0,\ \mathbf{x} \ge 0$$

where: f = the funds required for the building programme

P_m = the price of electricity in mode m

All other terms are the same as in problem (6.1)

The funds constraint (6.1 b) has been converted into the calculation of the funds required (6.2 b), which will be reported as part of the solution.

Expected Profit: 38.18										
Scenario (profit):		High (45)			Average (41)			Low (27.6)		
		Despatch			Despatch			Despatch		
Plant	Built	Base	Middle	Peak	Base	Middle	Peak	Base	Middle	Peak
P1	1	1	0	0	0	1	0	0	1	0
P2	2	1	1	0	0	2	0	0	0	2
P3	5	5	0	0	5	0	0	3	2	0
P4	0	0	0	0	0	0	0	0	0	0

Table 6.5: Optimal Expected Value Solution

The right hand sides of resource availability constraints, such as the funds constraint in this problem, are often somewhat arbitrary (or fuzzy). It is likely that there is a preferred level of investment, but that additional funds could be obtained if there were sufficiently good reason. A difficulty when formulating constraints of this sort is that the fuzzy availability of funds must be expressed as a single, hard number, and it is unclear what value should be used. Should it be the preferred level of investment, or the maximum possible? If the constraint is significant, it will be binding, and the decision maker is likely to be interested in the effect on the optimal solution of changing the RHS. When there is only one such constraint the shadow price will provide some information, but there will often be many such constraints, and interpretation of the shadow prices will be difficult.

When building a model for analysis using NSSA, these difficulties can be overcome by converting these resource constraints into the calculation of the quantity of the resource required by the solution. The required resource can then be reported as part of the solution, and the decision maker can consider the need for resources when comparing alternatives. It can be helpful to set an upper bound on the availability of the resource, because this may reduce the size of the feasible region, and thus the size of the noninferior set, and the number of alternatives that the decision maker must consider. In this example no upper bound on available funds has been included.

The example data is altered by including electricity prices for stage two. The price of base load electricity is 55, middle load is 60 and peak load is 130 per unit. The new optimal expected value solution is shown in Table 6.5.

The stochastic optimal solution builds sufficient plant to meet demand under the Low scenario, and leaves demand unmet under the other two scenarios. As demand

Scenario	Plant Size				Objective Values			Funds
Optimised	P1	P2	P3	P4	High	Average	Low	Used
High	0	3	7	2	58.0	29.6	-7.3	145
Average	1	2	5	2	48.0	44.0	15.6	116
Low	3	2	3	0	41.0	37.0	30.0	92
	Absolute Differences:				17.0	14.4	37.3	
	Percentage Differences:				29.3%	32.7%	124.3%	

Table 6.6: Scenario-Maximal Decisions

increases from the Low scenario to the High scenario, first the demand for Peak load, and then for Middle load, are not fully supplied.

The three scenario-maximal solutions, and their performances under all scenarios, are shown in Table 6.6. The greatest difference between the scenario-maximal objective function value, and that observed when the scenario-maximal decision for a different scenario is implemented, is 37.3 under the Low scenario. This is 124.3% of the best outcome for the Low scenario. The smallest difference between objective function values occurs under the Average scenario, and is 14.4, or 32.7% of the best outcome for that scenario. These numbers are clearly significant, and it is now worth while including the uncertainty in the formulation. Because we have removed the funds constraint, it can be seen that the scenario-maximal solutions for the High scenario requires a greater investment than the 120 originally specified as being available.

6.2.3 An Application of Noninferior Set Scenario Analysis

In this section we apply Noninferior Set Scenario Analysis to the modified version of the Louveaux and Smeers problem described in Section 6.2.2.

First, the scenario-maximal solutions are found. These are the solutions shown in Table 6.6, and they appear as solutions v1, v2 and v3 in Table 6.7. The fourth point, v4, is then found, and these four points are used to form an initial 3-D polytope in objective space. This polytope is the initial inner bound to the noninferior set. In general, the fourth point may turn out to be inferior, or noninferior. In this example it turns out to be inferior.

The lower bounds on the scenario objectives are set to be 10% smaller than the minimum value over the scenario-maximal solutions, and the outer bounding polytope is formed as:

$$\begin{aligned} 36.90 &\leq z_{High} \leq 58.0 \\ 26.64 &\leq z_{Average} \leq 44.0 \\ -8.03 &\leq z_{Low} \leq 30.0 \end{aligned}$$

The longest diagonal across the outer bounding polytope is 46.83. A tolerance of 5% is used for this example, which gives a maximum allowable error of 2.34. The XNISE algorithm produces the noninferior extreme points shown in Table 6.7. The funds required to implement the corresponding solutions are shown in Table 6.8. The full set of extreme points found by the algorithm are listed in Table A.1, in Appendix A.1, and full descriptions of the solutions at these points are given in Appendix A.1.2.

The noninferior extreme points characterise the approximation of the noninferior set. The approximation consists of the seven triangular faces, and two edges, shown in Table 6.9. The first two faces, f18 and f19, are adjacent and coplanar, as are the last two faces, f4 and f8. These two pairs of faces each form a non-triangular face in the noninferior set, and so the noninferior set really consists of five faces and two edges. These five faces have been labelled *a* to *e*.

The noninferior set is shown graphically in Figure 6.1, in which the noninferior extreme points are sorted by the objective function value of the High scenario. This diagram is an example of the value path with the noninferior faces shown, as discussed in Section 5.4.2. The noninferior faces are shown for the Average and

Solution	Objective Values			Scenario Weights			Plant Size			
Number	High	Average	Low	High	Average	Low	P1	P2	P3	P4
v1	58.00	29.60	-7.30	1.000	0.000	0.000	0.0	3.0	7.0	2.0
v2	48.00	44.00	15.60	0.000	1.000	0.000	1.0	2.0	5.0	2.0
v3	41.00	37.00	30.00	0.000	0.000	1.000	3.0	2.0	3.0	0.0
v6	49.00	39.80	23.90	0.488	0.185	0.327	1.0	1.0	6.0	0.0
v11	51.00	38.60	18.70	0.776	0.000	0.224	0.0	1.0	7.0	0.0
v12	53.00	42.80	9.90	0.523	0.359	0.118	1.0	3.0	6.0	0.0
v13	45.00	41.00	27.60	0.021	0.652	0.327	1.0	2.0	5.0	0.0
v16	47.00	39.80	26.40	0.481	0.000	0.519	0.0	2.0	6.0	0.0
v18	55.00	41.60	4.70	0.741	0.189	0.070	0.0	3.0	7.0	0.0
v19	47.00	41.00	25.10	0.327	0.473	0.200	2.0	1.0	5.0	0.0

Table 6.7: The Noninferior Extreme Points that Characterise the Noninferior Set
This table lists the noninferior extreme points found using a maximum allowable error of 5%. These points characterise the approximation to the noninferior set. The points are numbered in the order in which they were found by the algorithm. The complete set of extreme points that define the inner bounding polytope are shown in Table A.1, in Appendix A.1.

Noninferior Point	v1	v18	v12	v2	v11	v6	v16	v19	v13	v3
Total Plant Built	12	10	10	10	8	8	8	8	8	8
Total Construction Cost	145	133	127	116	119	113	110	107	104	92

Table 6.8:
The Construction Programmes Corresponding to the Noninferior Extreme Points
This table lists the total plant capacity to be built, and the cost of the building programme, for each noninferior extreme point.

Low scenarios. Because the band of noninferior values for the High scenario is too narrow to show the faces, the High scenario is drawn at a larger scale in Figure 6.2, with the faces shown. In Figure 6.2, face *a* cannot be seen because it lies "edge on" along the edge v6-v11-v12-v18. Similarly, face *d* lies along edge v13-v19-v2. Face *c* is hidden by face *b*.

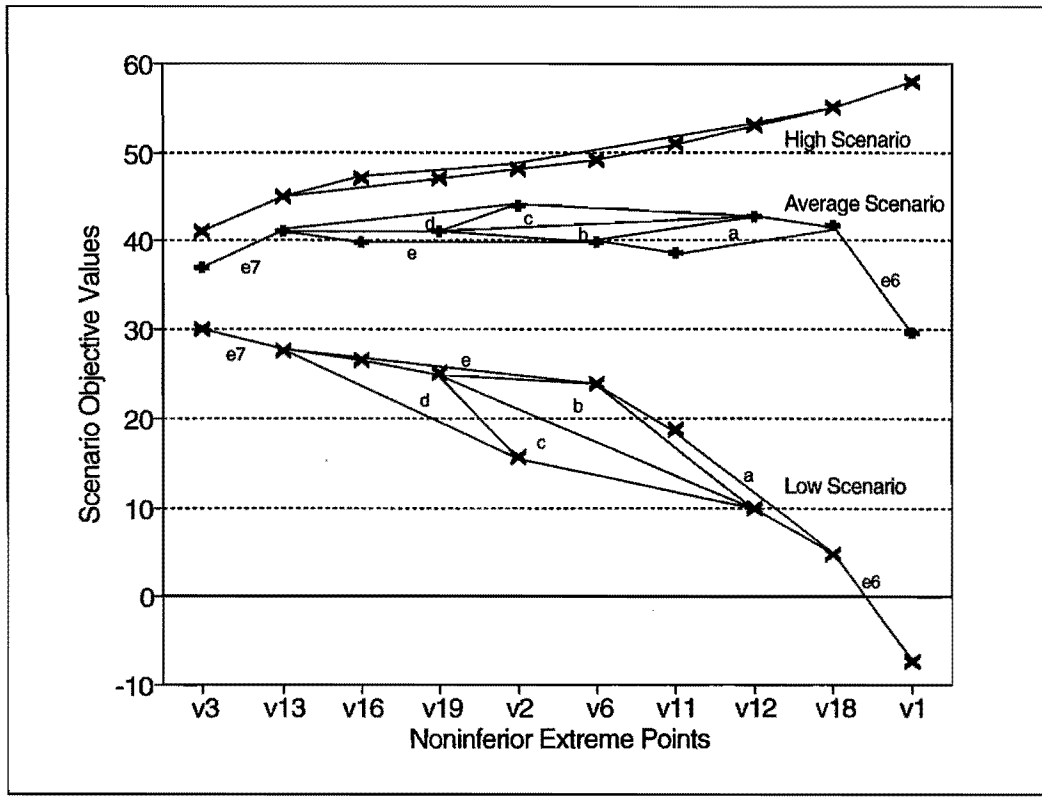

Figure 6.1: The Noninferior Set
The noninferior set displayed for each scenario individually. The noninferior faces are labelled a to e.

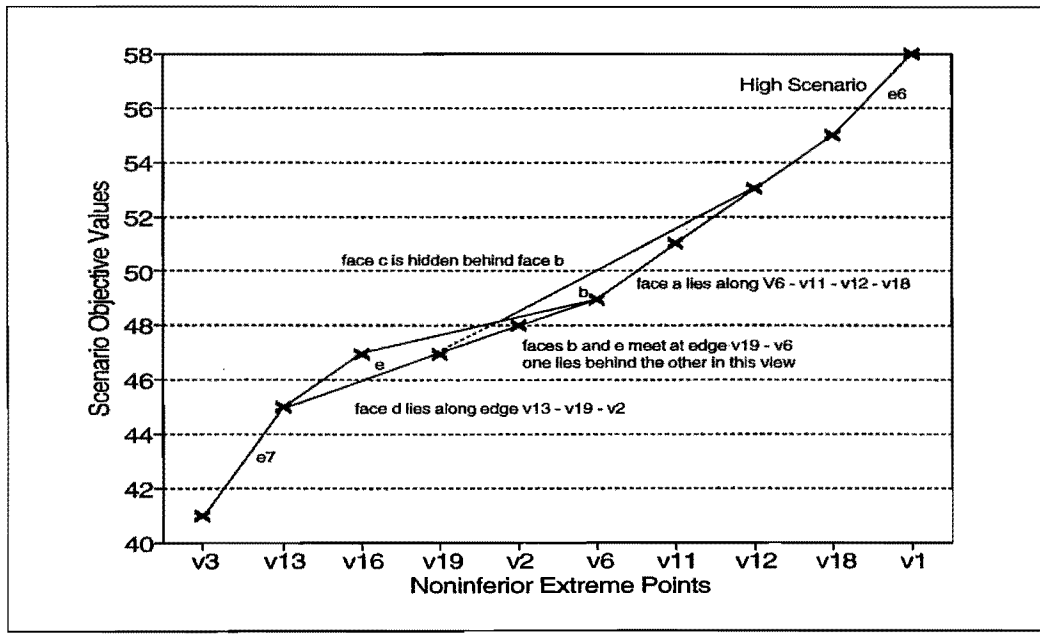

Figure 6.2: The Noninferior Set for the High Scenario Alone
The noninferior set displayed at a larger scale for the High scenario, so that the faces can be distinguished. The noninferior faces have the same labels as in Figure 6.1 above.

Faces	Vertices			Neighbouring Faces			Scenario Weights			Weighted Value	Label
f18	v18	v11	v6	f10	f19	f32	0.643	0.143	0.214	42.31	a
f19	v18	v12	v6	f13	f18	f31	0.643	0.143	0.214	42.31	a
f13	v19	v12	v6	f19	f4	f33	0.375	0.426	0.199	40.09	b
f33	v19	v12	v2	f17	f13	f34	0.329	0.483	0.187	40.00	c
f34	v19	v13	v2	f25	f8	f33	0.250	0.550	0.200	39.32	d
f4	v19	v16	v6	f2	f8	f13	0.375	0.325	0.300	38.48	e
f8	v19	v16	v13	f27	f4	f34	0.375	0.325	0.300	38.48	e
e6	v1	v18		f20			0.800	0.200	0.0	52.32	
				f32			0.800	-0.024	0.224	44.07	
e7	v3	v13		f26			0.	0.375	0.625	32.63	
				f27			0.375	0.0	0.625	34.13	

Table 6.9: Faces in the Approximation to the Noninferior Set
The approximation to the noninferior set is made up of seven faces and two edges. The faces have been labelled a to e, and these labels are used in Figures 6.1 and 6.2. Faces f18 and f19 are coplanar, and are given the same label, to show that they are in fact a single face. Similarly, faces f4 and f8 are coplanar and have been given a single label. The vertices of each face are shown, along with its neighbouring faces, and the weighting vector for which the face is the optimal solution to the weighting problem. Every point on a face corresponds to an alternative optimal solution for that weighting vector. The dot product of the objective vector and the weighting vector gives the weighted value. None of the faces have a zero maximum possible error, and so none of them are known to be noninferior. The faces on each side of the two edges are shown with their weighting vectors. The criterion vectors on the edges are optimal for any convex combination of the weights of the faces on each side of the edge.

The noninferior set is represented as a decision map in Figure 6.3. The objective function values for the High and Average scenarios are on the axes, with the objective function values for the Low scenario shown as contour lines. Because the slopes of the contour lines are different on the different faces, the edges between the faces are shown as dotted lines. For example, the contour $Z_L = 20$ crosses face a, face b, face c, and finally face d. This can also be seen in Figure 6.1, but Figure 6.1 does not provide the trade-off information given by the slopes of the contours in Figure 6.3.

As well as wanting to know the performances of the stage one decisions under the scenarios, the decision maker will want to know the investments required. This information is summarised in Figure 6.4 for the noninferior extreme points. The total plant capacity required, and the mixes of plant types are summarised on the bars, and the total investment cost is shown by the line. The value path (Figure 6.5) is shown beside the bar graph to show the relationships between the plant installed

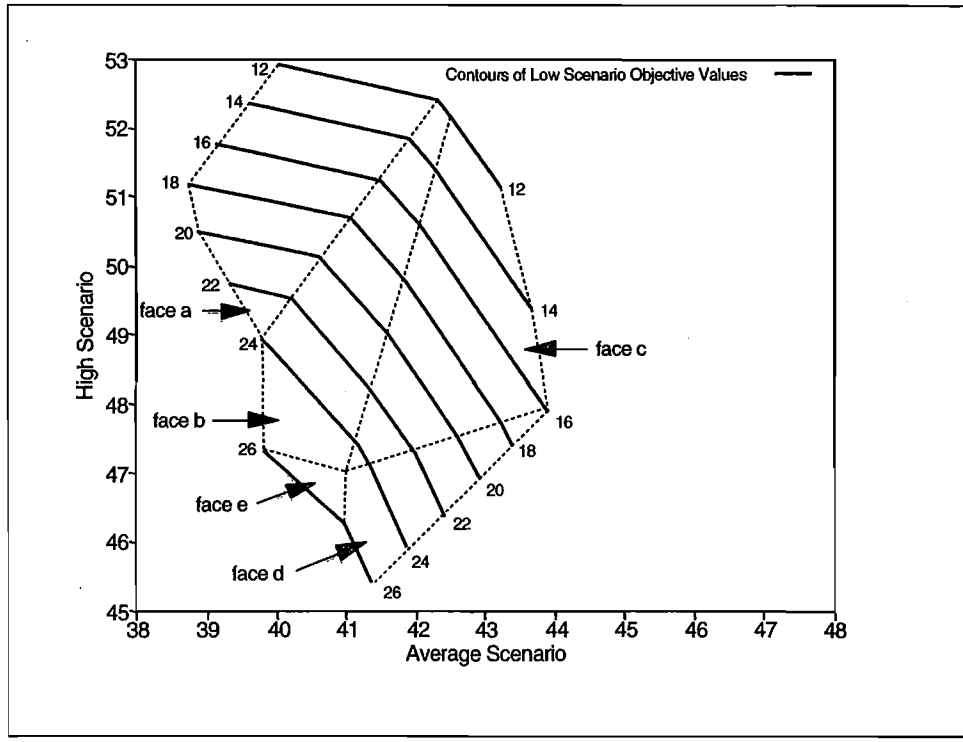

Figure 6.3: Decision Map with Contours for the Low Scenario
The slopes of the contour lines reflect the trade-offs between the scenarios. Because the slopes change as the contours cross from face to face, the edges where the faces meet are shown by the dotted lines.

and performance under the scenarios.

The noninferior set has been mapped onto the weighting space in Figure 6.6. As discussed in Section 5.4.4, each region in weighting space corresponds to an extreme point in criterion space, and the points in weighting space correspond to the faces in criterion space. The line between the regions of two extreme points in weighting space corresponds to the edge between the two extreme points in criterion space. The same names are used to label the extreme points and noninferior faces in Figure 6.6 and in Figures 6.1 and 6.2.

In Figure 6.7 the weighting space diagram is shown again with regions of interest shaded. In the left-hand graph, the extreme points that call for the installation of more than 8 units of plant are shaded. It is immediately apparent that installing more that 8 units of plant corresponds to placing a weight of no more than 0.2 on the Low scenario. This is consistent with Figures 6.4 and 6.5, from which it can be seen that the solutions that call for more than 8 units of plant perform poorly under the Low scenario. In the right-hand graph in Figure 6.7 the region in which

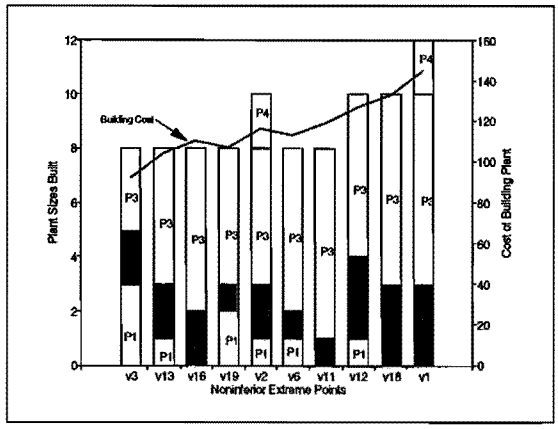

Figure 6.4:
Bar Graph of Plant Sizes
This graph shows the total plant capacities, and the mixes of plant types, that are specified by the solution at each of the noninferior extreme points.

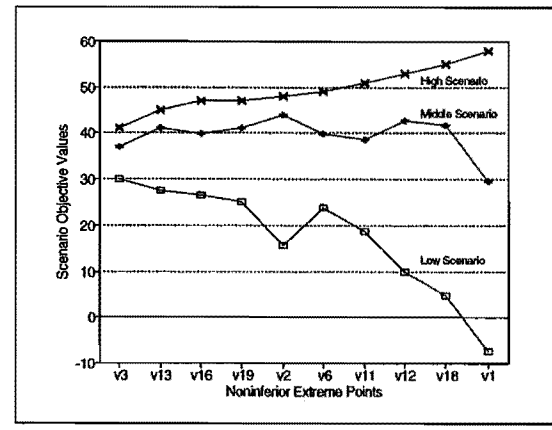

Figure 6.5:
Value Path of Scenario Objective Function Values
This graph shows the scenario objective function values for each noninferior extreme point.

all scenarios are given a weighting of at least 0.2 is shaded. This region is of interest to a decision maker who considers that all of the scenarios have a similar level of importance.

We will now simulate a decision maker using this output to select a decision to implement. On studying Figures 6.1 and 6.2 he decides that he is not willing to implement any decision that will return a profit of less than 10 under the Low scenario, or less than 40 under the Average scenario. Because extreme points v16 and v6 return profits of 39.8 under the Average scenario he decides to use 39.8 as the lower limit, so that they remain feasible. Figures 6.1 and 6.2 are modified to become Figure 6.8, by removing the regions that have been declared unacceptable. Three new points are shown in this figure, being E40 (43.8, 39.8, 28.3), L10 (53.49, 40.46, 10.00) and M40 (52.60, 39.80, 13.10). E40 is the point where the plane $z_A \geq 39.8$ cuts across the edge $e7$. L10 is the point at which the plane $z_L \geq 10$ cuts across the edge v11–v18 of face a, and M40 is the point at which the plane $z_A \geq 39.8$ cuts across the edge v11–v18 of face a. This reduction of the feasible region eliminates extreme points v3, v11, v18 and v1 from the noninferior set.

By looking at Figures 6.4 and 6.5 the decision maker sees that 6 of the 10 noninferior decisions build only 8 units of plant, which is sufficient to meet demand under the Low scenario, but not under the others. The decisions that build more than 8 units of plant (v2, v12, v18 and v1) perform rather poorly under the Low

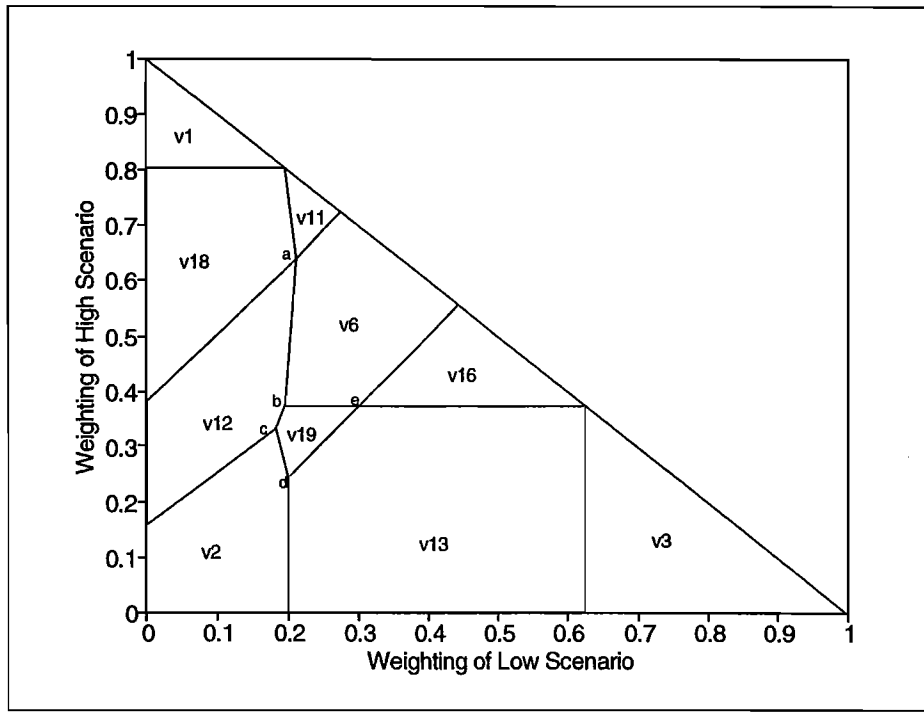

Figure 6.6: Noninferior Points Plotted in Weighting Space
The noninferior extreme points mapped onto the weighting space. Each combination of the weights placed on the scenarios is a point in weighting space that lies on the triangular plane $(w_H + w_A + w_l = L|\mathbf{w} \geq \mathbf{0})$. The optimal solution for any weighting of the scenarios can be read from the figure. For example, for the weighting vector $(w_H = 0.5, w_A = 0.2, w_L = 0.3)$ v6 is optimal.

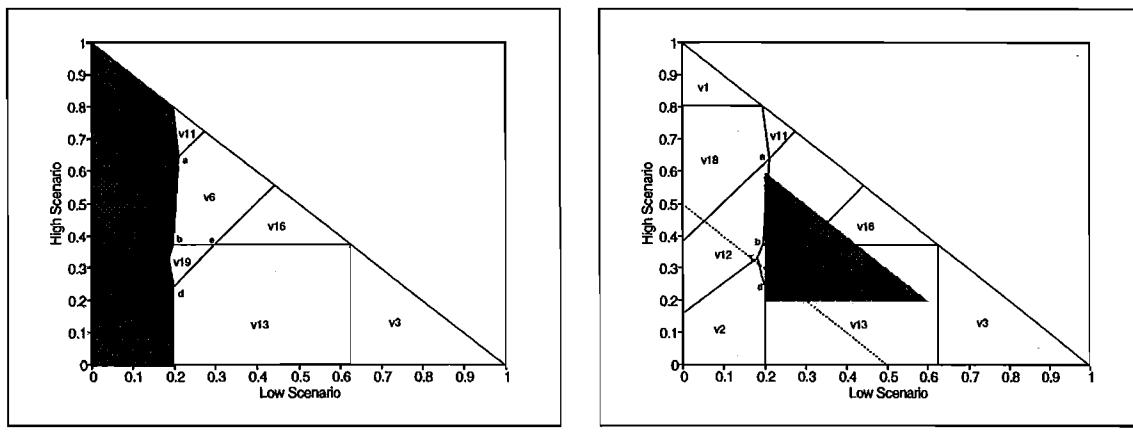

Figure 6.7:
Noninferior Points Plotted in Weighting Space with Certain Regions Shaded.
These graphs are redisplays of Figure 6.3 with regions of interest shaded. In the left-hand figure, the solutions that build more than the minimum of 8 units of plant are shaded. In the right-hand figure, the region within which all scenario weights are at least 0.2 is shaded.

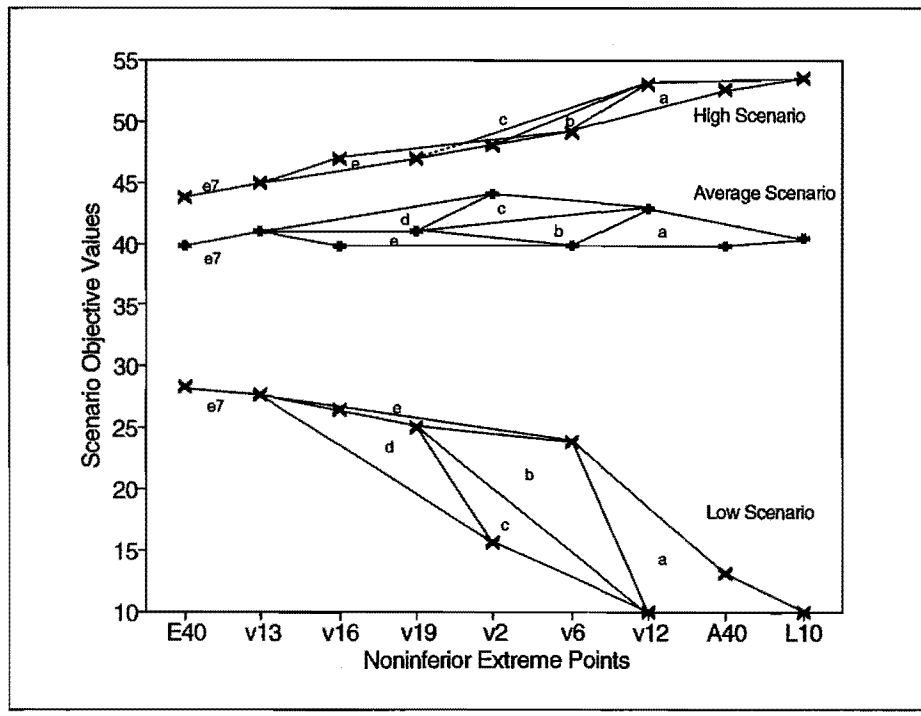

Figure 6.8: The Reduced Noninferior Set
The noninferior set of Figure 6.1 reduced by the restrictions that the profit under the Low scenario be at least 10, and under the Average Scenario be at least 39.8.

scenario. It is apparent from the behaviour of value path (Figure 6.5) that the decision to build sufficient plant to meet demand under the High scenario, (solution v1) is a high risk venture, because this decision sharply reduces the profit for both the Average and Low scenarios from the next best decision (v18).

The decision maker now moves to Figure 6.3. This decision map includes a region in which the profit of the Average scenario is less than 39.8, and so it is updated to become Figure 6.9.

After studying Figure 6.9, the decision maker decides that the profit under the High scenario should be 48, and under the Average scenario it should be 42, which places his preferred decision between contours 20 and 22 for the Low scenario. From the equation of face c $(0.329z_H, 0.483z_A, 0.187z_L = 40)$, (see Table 6.9), the profit under the Low scenario is found to be 20.97, and the decision maker's preferred criterion vector is: (48, 42, 20.97).

This criterion vector is translated into a solution vector by finding the convex combination operator, γ, that calculates the criterion vector from the corner points

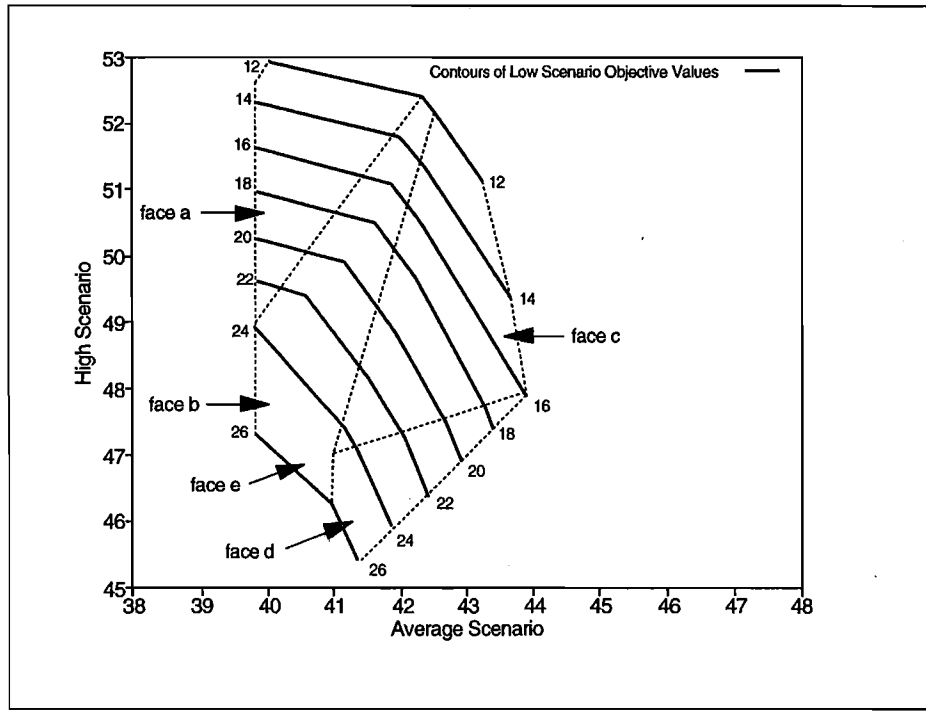

Figure 6.9: The Reduced Decision Map
The decision map of Figure 6.3 with the region z(Average) $\leq$ 39.8 removed.

of face *c*. γ is found by solving the equation set:

$$[\mathbf{v19} \,|\, \mathbf{v12} \,|\, \mathbf{v2}]\gamma = (48, 42, 20.97)^T \tag{6.2}$$

which gives $\gamma^T = (0.634, 0.117, 0.249)$

The solution vector for the preferred criterion vector is found by solving the equations:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y}_H \\ \mathbf{y}_A \\ \mathbf{y}_L \end{bmatrix} = \begin{matrix} \text{v19} & \text{v12} & \text{v2} \\ \end{matrix} \begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{y}_H & \mathbf{y}_H & \mathbf{y}_H \\ \mathbf{y}_A & \mathbf{y}_A & \mathbf{y}_A \\ \mathbf{y}_L & \mathbf{y}_L & \mathbf{y}_L \end{bmatrix} \gamma$$

where: $\mathbf{x}$ = the stage one vector for each criterion vector

$\mathbf{y}_H$ = recourse vector under the High scenario

$\mathbf{y}_A$ = recourse vector under the Average scenario

$\mathbf{y}_L$ = recourse vector under the Low scenario

This produces the solution shown in Table 6.10.

Scenario (profit)		High (48) Despatch			Average (42) Despatch			Low (20.97) Despatch		
Plant	Built	Base	Middle	Peak	Base	Middle	Peak	Base	Middle	Peak
P1	1.634	1.634	0	0	0	1.634	0	0	0.883	0.751
P2	1.483	0.249	1.234	0	0	1.249	0.234	0	0	1.249
P3	5.117	5.117	0	0	5.000	0.117	0	3.000	2.117	0
P4	0.498	0	0	0.498	0	0	0.498	0	0	0
Totals:	8.732	7.000	1.234	0.498	5.000	3.000	0.732	3.000	3.000	2.000
Unmet Demand:		0	1.766	1.502	0	0	1.268	0	0	0

Table 6.10: Calculated Solution for the Preferred Criterion Vector
This solution builds 0.765 units of plant that will be used under the High and Average scenarios, but will be left unused under the Low scenario. Demand will be left unmet if either of the High or Average scenarios occur.

The decision maker decides that this is an acceptable solution, but notes that face c is not flagged as being known to be noninferior. This means that the solution in Table 6.10 may be dominated. Because he chose the criterion vector by setting profit levels for the High and Average scenarios, and then finding the corresponding profit for the Low scenario, he decides to improve the profit under the Low scenario, if possible. This is done by solving problem (6.2) with the addition of the constraints:

$$z_H \geq 48$$

$$z_A \geq 42$$

and using the weighting vector $\mathbf{p} = (0, 0, 1)$. This maximises the profit under the Low scenario, while maintaining the desired profit levels under the other two scenarios. The resulting solution, shown in Table 6.11, builds a different pattern of plant types, with a slightly lower total capacity, and a slightly improved profit under the Low scenario.

As discussed in Section 5.5, another way of finding an efficient solution from a preferred criterion vector is to set lower bounds on the scenario objective function

Scenario (profit)		High (48) Despatch			Average (42) Despatch			Low (21.1) Despatch		
Plant	Built	Base	Middle	Peak	Base	Middle	Peak	Base	Middle	Peak
P1	2	2	0	0	0	2	0	0	1	1
P2	1	0	1	0	0	1	0	0	0	1
P3	5	5	0	0	5	0	0	3	2	0
P4	0.667	0	0	0.667	0	0	0.667	0	0	0
Totals:	8.667	7	1	0.667	5	3	0.667	3	3	2
Unmet Demand:		0	2	1.333	0	0	1.333	0	0	0

Table 6.11:
Preferred Solution and Criterion Vector Found by Resolving the Problem
This solution builds 0.667 units of plant that will be used under the High and Average scenarios, but will be left unused under the Low scenario. Demand will be left unmet if either of the High or Average scenarios occur. This solution produces a slightly higher profit under the Low scenario than that of the preferred solution found from the approximation to the noninferior set. See Table 6.10.

values and use new criteria to form the objective. For this example, it would be reasonable for the decision maker to want to achieve his preferred criterion vector while minimising the stage one investment. This could be done by solving problem (6.2) with the objective function: minimise $\sum_{g=1}^{4} bc_g x_g$ and the addition of the constraints:

$$z_H \geq 48$$
$$z_A \geq 42$$
$$z_L \geq 20.97$$

This formulation also returns the optimal solution shown in Table 6.11.

Another approach is to refer to Figure 6.7 to see which decisions are optimal when different weights are placed on the scenarios. For example, if a weight of 0.5 is placed on the Average scenario, the solutions along the dotted line in Figure 6.7 are optimal in turn as the weight placed on the High scenario changes from 0.5 to 0 (the weight on the Low scenario changes from 0 to 0.5). That is, decisions v18, v12, v19, and v13.

If the scenarios are weighted equally, then v13 is optimal, and v19, v6 and v16 are almost optimal. If the decision maker wants all scenarios to have a weight of at least 0.2, then the choice is reduced to decisions v6, v16, v13 and v19, as shown by the shaded area in the right-hand diagram.

In this example, the decision maker selected a solution from face c, which is just

outside the shaded region, and has a weighting vector of (0.33, 0.48, 0.19).

6.2.4 The Use of Different Maximum Allowable Errors

The maximum allowable error was set rather arbitrarily at 5% of the longest diagonal across the initial inner bound. The problem was also solved using 15%, 10% and 0% to calculate the maximum allowable error. The noninferior faces found using these four maximum allowable errors are shown in Figures 6.10 to 6.13, and the faces are listed in Tables 6.12 to 6.14. The extreme points, and the faces, have been labelled so that a given label refers to the same point, or face, in every diagram.

Faces	Vertices			Neighbouring Faces			Scenario Weights			Weighted Value
f10	v18	v6	v2	f4	f8	f17	0.406	0.427	0.167	40.88
f4	v6	v3	v2	f10	f13	f2	0.021	0.652	0.327	34.80
e1	v1	v18		f18			0.800	0.200	0.0	52.32
				f7			0.800	-0.028	0.228	43.90

Table 6.12: Faces in the Approximation to the Noninferior Set with mae = 15%
The noninferior faces in the approximation when the problem is solved with the maximum allowable error (mae) set to 15%.

Faces	Vertices			Neighbouring Faces			Scenario Weights			Weighted Value	Label
f18	v12	v11	v6	f10	f19	f32	0.643	0.143	0.214	42.31	a
f26	v18	v12	v11	f18	f20	f25	0.643	0.143	0.214	42.31	a
f13	v13	v6	v2	f8	f2	f21	0.327	0.473	0.200	39.63	
f8	v12	v6	v2	f13	f17	f18	0.336	0.468	0.196	39.77	
e4	v1	v18		f25			0.780	-0.024	0.224	44.07	
				f19			0.800	0.200	0.0	52.32	
e5	v3	v13		f22			-0.574	0.949	0.625	30.33	
				f5			0.375	0.0	0.625	34.13	

Table 6.13:
Faces in the Approximation to the Noninferior Set with mae = 10%
The noninferior faces in the approximation when the problem is solved with the maximum allowable error (mae) set to 10%.

The sequence of figures from 6.10 to 6.13 illustrates how the detail with which the noninferior set is described increases as the maximum allowable error decreases. When the mae = 15% the noninferior set is approximated by only two faces and one edge. When the mae = 10%, the two faces have been replaced by an edge and three

Faces	Vertices			Neighbouring Faces			Scenario Weights			Weighted Value	Label
f18	v18	v11	v6	f10	f19	f32	0.643	0.143	0.214	42.31	a
f19	v18	v12	v6	f33	f36	f18	0.643	0.143	0.214	42.31	a
f35	v29	v23	v12	f36	f33	f51	0.375	0.438	0.188	40.46	b
f51	v29	v28	v23	f31	f35	f54	0.375	0.438	0.188	40.46	b
f53	v29	v19	v6	f4	f33	f54	0.375	0.425	0.2	40.07	b
f54	v29	v28	v19	f52	f51	f53	0.375	0.425	0.2	40.07	b
f52	v28	v19	v2	f54	f34	f1	0.25	0.55	0.2	39.32	d
f34	v19	v13	v2	f52	f25	f8	0.25	0.55	0.2	39.32	d
f8	v19	v16	v13	f29	f4	f34	0.375	0.325	0.3	38.48	e
f4	v19	v16	v6	f53	f2	f8	0.375	0.325	0.3	38.48	e
f33	v29	v12	v6	f19	f35	f53	0.4	0.4	0.2	40.30	f
e10	v1	v18		f43			0.8	0.0	0.2	44.94	
				f20			0.8	0.2	0.0	52.32	
e11	v3	v13		f39			0.0	0.375	0.625	32.63	
				f28			0.375	0.0	0.625	34.13	

Table 6.14: Faces in the Approximation to the Noninferior Set with mae = 0%
The noninferior faces in the approximation when the problem is solved with the maximum allowable error (mae) set to 0%.

faces, while edge *e1* (now labelled *e4*) remains unchanged. Additional faces are added when the mae = 5%, but when the mae = 0% the number of faces reduces to four. Table 6.14, which lists the faces found with mae = 0%, shows eleven triangular faces, but many of these are coplanar (or almost coplanar), and it is the combined, non-triangular, faces that are shown in Figure 6.13.

For this example, the approximations found with mae = 15%, and mae = 10%, do not give a good picture of the noninferior set, but the approximation found with mae = 5% is very close to that found using mae = 0%. In fact, it would seem that little is gained by going from 5% to 0%.

Figures 6.14 and 6.15 show the decision map found with mae = 15% and the decision map found with mae = 5%. Clearly the decision map with mae = 15% does not give a good picture of the trade-offs. In Figure 6.14, the contours on face f4 are almost vertical, which means that the trade-offs are so severe that the decision maker would be most unlikely to select a solution from that face. However, to find the preferred solution chosen by the decision maker in Section 6.2.3: $(48, 42, 20.76)$,

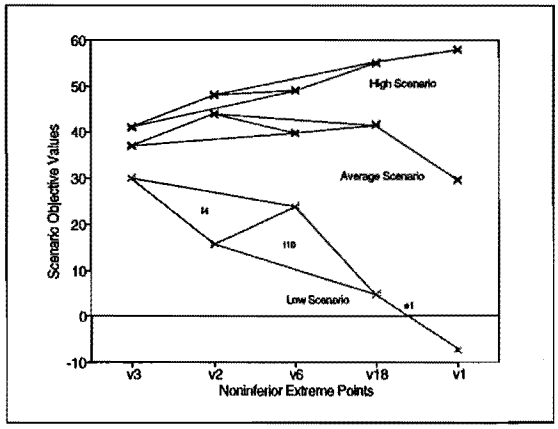

Figure 6.10:

Noninferior Set with mae = 15%

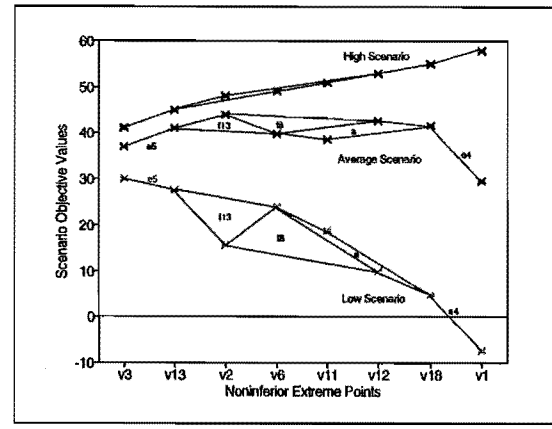

Figure 6.11:

Noninferior Set with mae = 10%

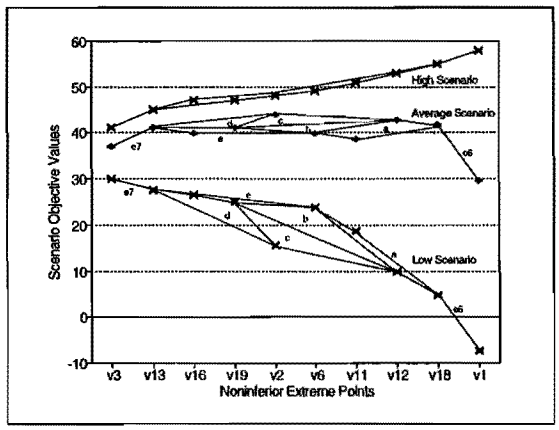

Figure 6.12:

Noninferior Set with mae = 5%

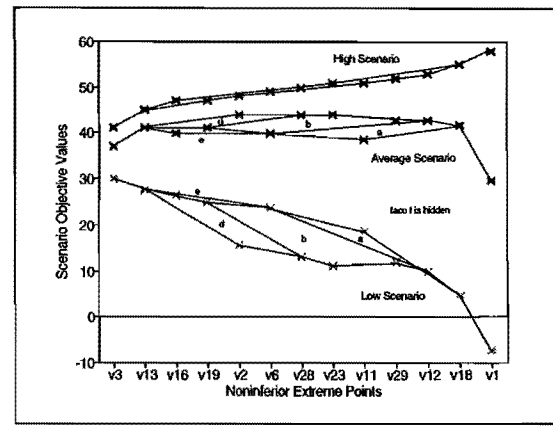

Figure 6.13:

Noninferior Set with mae = 0%

a criterion vector would have to be chosen from this face.

These comparisons suggest that a satisfactory solution is unlikely to be found if the approximation to the noninferior set is found with a large maximum allowable error. However, it would be possible to start with a large maximum allowable error, and use that approximation to determine bounds on the scenario objectives. These bounds would reduce the size of the approximation to the noninferior set found when a smaller mae is used, and thus reduce the computational effort. In this example, the approximation with mae = 15% was used to determine that the minimum acceptable objective function values under the Average and Low scenarios are 40 and 10 respectively. The problem was then solved to find an approximation

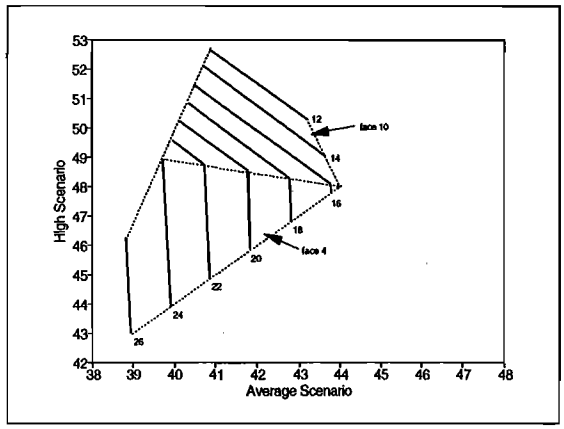

Figure 6.14:

Decision Map with mae = 15%
The decision map found when the maximum allowable error (mae) is set to 15%. The vertical contour lines show that, in this approximation, face f4 is very close to being dominated.

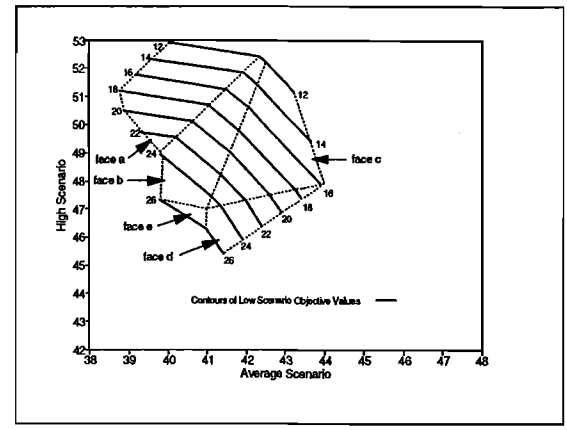

Figure 6.15:

Decision Map with mae = 5%
The High Scenario axis has been changed from Figure 6.3 to match the axis in Figure 6.14.

to the noninferior set with the mae = 5% and a lower bound on the objective of the Average scenario of 39.8, and on the Low scenario of 10. (The bound of 39.8 was used to facilitate comparisons with the results obtained when the size of the noninferior set was reduced in Section 6.2.3.) The resulting approximation to the noninferior set is shown in Figure 6.16, with the one found in Section 6.2.3 beside it. The overall shapes of the two approximations are similar, but the approximations are described by different sets of faces. This is to be expected because the presence of the lower bounds on the Average and Low scenarios will have made the outer bounding polytope smaller than when the problem was solved without the bounds. This means that the maximum possible errors will have been different, and the faces will have been selected in a different order to find new extreme points.

As the maximum allowable error is reduced, the computational effort required to find the approximation increases. Summary statistics for the four mae values used here are listed in Table 6.15. The problem "Scenarios" is the problem being solved, that is, problem (6.2). Problem "HUZ" finds the maximum possible error for each face in the approximation, and is problem (5.4), on page 125. Problem "Check" identifies the faces and edges that form the approximation, and checks for any inferior criterion vectors. These are problems (5.5) and (5.6), on pages 136

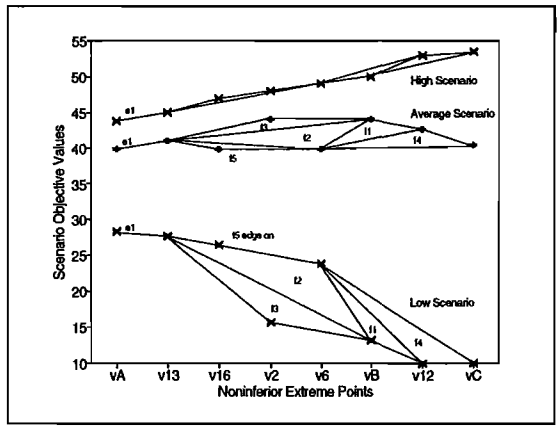

Figure 6.16:
Noninferior Set Found with Lower Bounds on Objectives.
The noninferior set found with the constraints that the profit under the Low scenario be at least 10, and under the Average Scenario be at least 39.8.

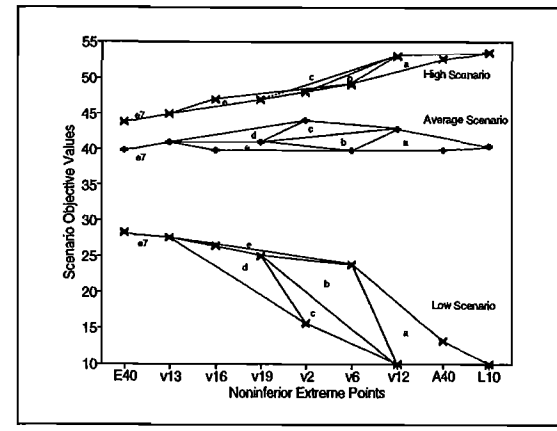

Figure 6.17:
The Reduced Noninferior Set
The noninferior set found from the noninferior set of Figures 6.1 and 6.2 when they are pruned with the restrictions that the profit under the Low scenario be at least 10, and under the Average Scenario be at least 39.8.

and 138 respectively. The column "Calls" shows the number of times that CPLEX was called to solve an instance of the problem. The column "Iterations" shows the number of iterations used by CPLEX, summed over all of the calls.

	Problem Scenarios		Problem HUZ		Problem Check		Elapsed Time
mae	Calls	Iterations	Calls	Iterations	Calls	Iterations	seconds
15%	13	141	30	184	6	38	20
10%	18	184	49	384	6	74	28
5%	23	180	78	566	6	53	38
0%	96	2060	160	1004	4	70	94

Table 6.15: Summary Resource Statistics for Different Settings of the mae.
Problem 'scenarios' is the problem being solved. HUZ finds the maximum possible error for each face in the approximation. Problem 'Check' identifies the faces that should be included in the approximation, and checks for inferior criterion vectors in the approximation. The number of times CPLEX was called, and the total number of iterations summed over all of the calls are also shown.

The statistics in Table 6.15 suggest that the increase in computational effort is not great as the maximum allowable error is reduced, provided it is not reduced to zero. When it is reduced to zero, the noninferior set is found exactly, and the computational effort increases considerably. For the example problem solved here, the approximation to the noninferior set was quite adequate when the maximum

allowable error was set to 5%, and little is gained by expending the additional effort required to find the noninferior set exactly.

Large problems may have thousands of noninferior extreme points, in which case the choice of the maximum allowable error will be very important. One of the strengths of the XNISE algorithm is that it can find an approximation to the noninferior set without finding all of the noninferior extreme points. If a problem has thousands of noninferior extreme points, XNISE can be expected to describe an approximation to the noninferior set using a very small proportion of these points, provided the maximum allowable error has been set to a large enough value. Clearly, when the maximum allowable error is set to zero, all noninferior points will be found, and this strength of the XNISE algorithm is wasted.

6.3 Problem APL1P

The second example is test problem, APL1P, taken from Infanger (1994). It is a two-stage model of a simple power network in which the objective is to meet demand at minimum cost. Two types of generating plant can be built, with different investment and operating costs, and the demand is represented as a load duration curve with three modes: base, medium and peak. Generation capacity is to be installed in stage one to meet an uncertain demand in stage two. The stage two demand can be met by any combination of operating the generating plant built in stage one, and/or purchasing electricity from other sources. In addition to the uncertainty about demand there is uncertainty about the proportion of the installed capacity that will actually be available to produce electricity.

Because any quantity of electricity can be bought the problem has complete recourse. The data is summarised in Table 6.16, and formulated as problem (6.3).

Source of Supply	G1	G2	Buy		
Minimum Capacity (MW)	1000	1000	–		
Building Cost ($\$10^5$/MW)	4.0	2.5	–		
Production Costs ($\$10^5$/MW)					
Supply Base Load	4.3	8.7	10.0		
Supply Mid Load	2.0	4.0	10.0		
Supply Peak Load	0.5	1.0	10.0		
Demands (MW) These are the same for the three loads.					
Outcome	900	1000	1100	1200	
Probability	0.15	0.45	0.25	0.15	
Availability of Generators					
G1					
Outcome	1.0	0.9	0.5	0.1	
Probability	0.2	0.3	0.4	0.1	
G2					
Outcome	1.0	0.9	0.7	0.1	0.0
Probability	0.1	0.2	0.5	0.1	0.1

Table 6.16: Model APL1P: Test Problem Data

$$\text{minimise} \qquad Z = \sum_{\omega \in \Omega} p^{\omega} z^{\omega} \tag{6.3a}$$

$$\text{subject to} \qquad \sum_{i=1}^{2} c_i x_i + \sum_{i=1}^{3} \sum_{j=1}^{3} f_{ij} y_{ij}^{\omega} = z^{\omega} \qquad \omega \in \Omega \tag{6.3b}$$

$$x_i \geq b_i \qquad i = 1,2 \tag{6.3c}$$

$$-\alpha_i^{\omega} x_i + \sum_{j=1}^{3} y_{ij}^{\omega} \leq 0 \qquad i = 1,2 \qquad \omega \in \Omega \tag{6.3d}$$

$$\sum_{i=1}^{3} y_{ij}^{\omega} \geq D_j^{\omega} \qquad j = 1,2,3 \quad \omega \in \Omega \tag{6.3e}$$

$$x_i \geq 0 \qquad i = 1,2$$

$$y_{ij}^{\omega} \geq 0 \qquad i = 1,2,3 \quad j = 1,2,3 \quad \omega \in \Omega$$

where: Ω = the set of scenarios

Z = the weighted objective function value

z^{ω} = the objective function value for scenario ω

p^{ω} = the weight placed on the objective function of scenario ω

x_i = the size to build of plant i (source 3 is purchase, and not built)

c_i = the unit cost of building plant i

α_i^{ω} = the availability of plant i, under scenario ω

b_i = the minimum allowed size for plant i

y_{ij}^{ω} = the despatch from source i to load j under scenario ω

f_{ij} = the cost of using source i to supply load j

D_j^{ω} = demand for load j under scenario ω

In Infanger (1994), the probabilities of the individual uncertain events are assumed to have the values shown in Table 6.16. The uncertainty is modelled as a set of scenarios, one for every possible combination of the values of the random variables. This produces 1280 scenarios, with probabilities that range from 0.00003 to 0.01823. Problem (6.3) is solved to find the solution with the lowest expected cost. Infanger uses this problem to demonstrate importance sampling, and to trial different sample sizes.

The approach taken by noninferior set scenario analysis is to summarise the

uncertainty as a small number of contrasting scenarios, and to solve the resulting small problem many times to find a set of noninferior decisions. Thus, in noninferior set scenario analysis a small problem is solved many times to find several solutions, whereas in stochastic optimisation a large problem is solved once to find one solution.

To carry out noninferior set scenario analysis, a small number of scenarios must be built from the available data. These scenarios should represent the principal concerns of the decision maker. In this example, the decision maker's objective is to meet demand at minimum cost. She will be concerned about high levels of demand forcing her to buy expensive power from outside, but she will also want to avoid leaving capacity unused if demand turns out to be low. Similarly, large plant capacities are required to meet demand when plant availabilities are low, but high availabilities would leave capacity unused. Clearly, these two sources of uncertainty interact. When demand and plant availability are both high, or both low, the two outcomes offset one another, whereas, when one is high and the other low, the effects compound.

Because the two plant types have different cost structures, the sizes of the two plants should be matched to the types of demand. Table 6.17 shows that, if plant availabilities are expected to be high, G1 should be built to supply the base and mid loads, and G2 should be built to supply the peak load. If availabilities are expected to be high for one plant, and low for the other, then the preferred plant/load combinations will change. However, G2 should never be built to supply base load, because it is always more expensive than the alternative of meeting base load by buying power. If the availability of either type of plant is expected to be very low, then it is cheaper to buy power than it is to build that type of plant.

Clearly, there is no building programme that matches all of the possible combinations of events, and the decision maker cannot provide exactly for all of them. She must either take a chance and build plant for a particular outcome (e.g. high availability and high demand) or she must choose a compromise decision that balances between all, or at least some, of the possible outcomes.

We will represent the decision maker's concerns by building three scenarios. The first two scenarios will contrast situations in which the demand for power is high, and the availability of one of the plants is high, and the other is low. Under Scenario LowG1, the plant availabilities are G1 = 0.5 and G2 = 0.8, while under

		Total Costs (10^5\$/MW)		
	Availability	Base	Mid	Peak
Plant G1	1.0	8.30	6.00	4.50
	0.9	8.74	6.44	4.94
	0.5	12.30	10.00	8.50
	0.1	44.30	42.00	40.50
Plant G2	1.0	11.20	6.50	3.50
	0.9	11.48	6.78	3.78
	0.7	12.27	7.57	4.57
	0.1	33.70	29.00	26.00
	0.0	–	–	–
Break Even Availabilities Against Buy-in Cost of 10				
Plant G1		0.70	0.50	0.42
Plant G2		–	0.42	0.28

Table 6.17:
Model APL1P: Total Unit Supply Costs for Each Plant Availability
This table shows the total cost (building plus operating) of using each plant type to supply each type of demand for the given plant availabilities. These figures show that plant G1 should be built to supply base load if its availability is expected to be 0.7 or better, and that G2 should never be built to supply base load. They also show that, at high availabilities, plant G2 is preferred to meet peak load, and that either plant is suitable for meeting mid load.

Scenario LowG2, the plant availabilities are G1 = 0.9 and G2 = 0.4. The availabilities are asymmetric to reflect the fact that G1 is more reliable than G2. Under both of these scenarios, demand is high (base and peak = 1100, mid = 1200). The third scenario, Scenario High, is a hedge against having excess capacity when plant availabilities are high and demand is low. In Scenario High, the availabilities of both of plants G1 and G2 are 0.9, and the demand for both base and peak load is 1000, and for mid load is 900.

In the original formulation, a lower bound of 1000 MW is placed on both types of plant, although in the expected value solution neither bound is binding. In the NSSA formulation these lower bounds have been set to zero to avoid arbitrarily limiting the choices presented to the decision maker. The use of bounds to restrict the size of the noninferior set reduces the computational effort required to characterise it, but bounds also reduce the range of alternatives presented to the decision maker. However, reducing the range of alternatives presented to the decision maker can help reduce the problem of information overload. The important thing is to ensure that

Solutions	Objective Values ($B)			Plant Sizes (GW)		Plant
	LowG1	LowG2	High	G1	G2	Cost
v13	3.235	2.342	1.949	3.222	0.0	1.289
v2	3.235	2.279	2.171	3.778	0.0	1.511
v15	2.934	2.378	2.164	3.222	0.861	1.504
v3	2.846	2.514	1.832	2.111	1.111	1.122
v21	2.842	2.406	2.273	3.167	1.375	1.611
v20	2.691	2.550	2.345	2.400	2.750	1.648
v7	2.663	2.580	1.922	1.769	1.769	1.150
v14	2.584	2.774	1.862	1.111	2.111	0.972
v22	2.568	2.671	2.071	1.333	2.750	1.221
v19	2.556	2.690	2.090	1.278	2.875	1.230
v1	2.409	3.099	2.089	0.0	2.875	0.719

Table 6.18:
Noninferior Extreme Points that Characterise the Approximation
The coordinates in criterion space of the noninferior extreme points in the approximation, and the plant sizes, and construction costs, of the stage one decisions.

arbitrary bounds do not exclude alternatives of interest to the decision maker. In this example it turns out that lower bounds of 1000 MW would not have excluded the decision finally chosen by the decision maker. However, they would have reduced the available insights.

	Noninferior Extreme Points										
Scenario	v13	v2	v15	v3	v21	v20	v7	v14	v22	v19	v1
LowG1	1789	1511	1100	1456	717		1100	1156	533	461	1100
LowG2	500		156	1056		1100	1100	1556	1100	1100	2250
High											313

Table 6.19:
Power Bought under Each Scenario at the Noninferior Extreme Points
The quantities of power that must be bought to meet demand under each of the scenarios. The detail of which mode is supplied by buying power can be read from the full descriptions of the solutions in Appendix A.2.2.

The problem was solved to find an approximation to the noninferior set with the maximum allowable error set at 5% of the longest diagonal across the initial tetrahedron formed by the first four extreme points found. The results are summarised in Tables 6.18, 6.19 and 6.20, and presented graphically in Figures 6.20 to 6.23. In Figures 6.20 to 6.19 the noninferior extreme points are sorted by the objective

Faces	Vertices			Neighbouring Faces			Scenario Weights LowG1	LowG2	High	Weighted Value
f1	v15	v13	v3	f21	f3	f6	0.2085	0.6004	0.1911	2.453
f2	v22	v20	v7	f37	f23	f39	0.4885	0.5071	0.0044	2.618
f3	v15	v13	v2	f36	f31	f1	0.2006	0.6233	0.1761	2.452
f6	v15	v7	v3	f25	f10	f1	0.2803	0.5629	0.1568	2.500
f10	v14	v7	v3	f39	f6	f13	0.3061	0.2580	0.4359	2.319
f25	v21	v15	v7	f6	f36	f37	0.3145	0.5650	0.1205	2.527
f33	v19	v14	v1	f40	f30	f7	0.6178	0.2222	0.1600	2.511
f36	v21	v15	v2	f3	f25	f26	0.2432	0.7418	0.0150	2.510
f37	v21	v20	v7	f2	f25	f38	0.4870	0.5084	0.0046	2.618
f39	v22	v14	v7	f10	f2	f40	0.5441	0.2762	0.1798	2.507
f40	v22	v19	v14	f33	f34	f39	0.6105	0.2282	0.1613	2.511

Table 6.20: Faces in the Approximation to the Noninferior Set
The approximation to the noninferior set is made up of eleven faces. The vertices of each face are shown, along with its neighbouring faces, and the weighting vector for which the face is the optimal solution to the weighting problem. Every point on a face corresponds to an alternative optimal solution for that weighting vector, and the dot product of the objective vector and the weighting vector gives the weighted value. None of the faces have a zero maximum possible error, and so none of them are known to be noninferior.

function value under scenario LowG1. A full list of the solutions corresponding to the extreme points is included in Appendix A.2.2.

From Figures 6.20 and 6.21 it can be seen that the line showing the total construction cost generally has the same shape as the line showing the cost under Scenario High. This is consistent with the intuitive result that increasing the total capacity built will increase the unused the capacity when all plants have high availabilities. The exception is point v1, which reduces the construction cost from point v19 without reducing the cost under scenario High. Under solution v1 only plant G2 is built, which means that G2 must be used to meet the base load under Scenario High, although it would have been cheaper to build a smaller G2 plant and buy power to meet the base load. (Once G2 has been built, it is then cheaper to use it to meet base load, than it is to buy power.)

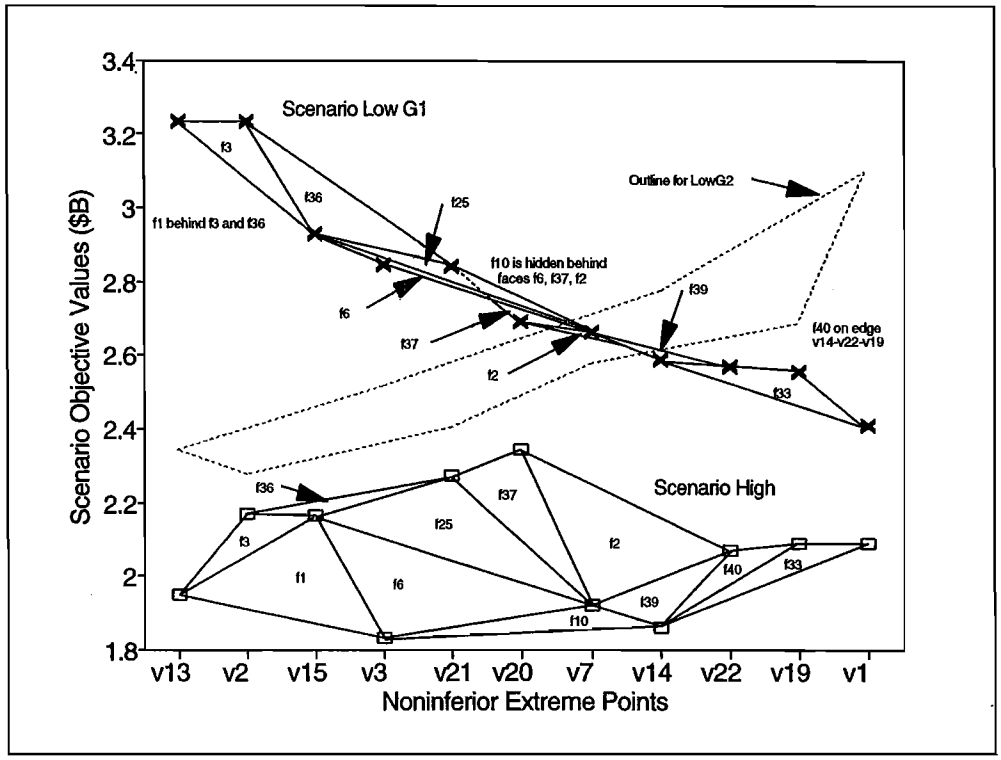

Figure 6.18: The Noninferior Set

The noninferior set displayed for each scenario individually. The noninferior faces are labelled. For the LowG1 scenario some of the faces cannot be seen in this representation, either because they are behind other faces, or because they are "edge on". The "shadow" of the noninferior set, as drawn for scenario LowG2, is included to show the tradeoffs between it and the other scenarios.

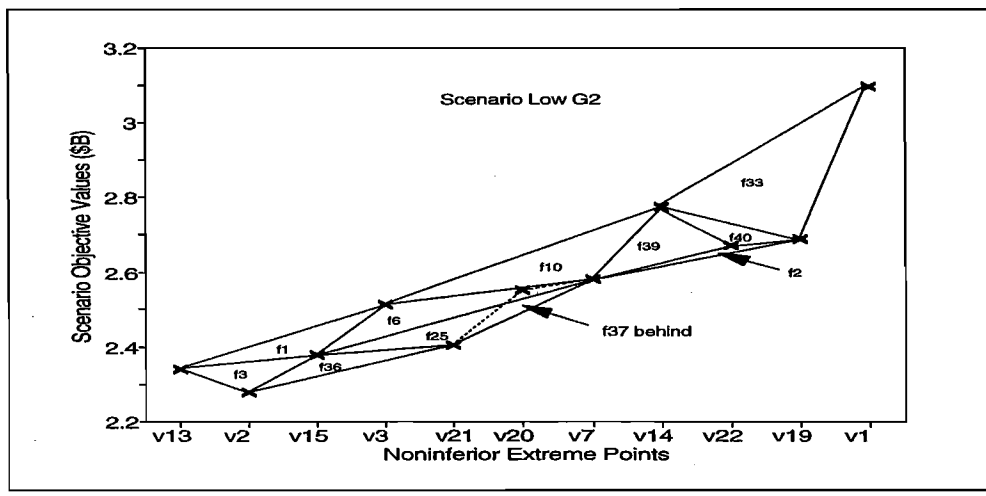

Figure 6.19: The Noninferior Set Drawn for Scenario LowG2

The noninferior set displayed separately for Scenario LowG2, because its value path crosses the value path of scenario LowG1, and this would make it confusing to display them both on the same diagram. This graph has been scaled to make distances on its vertical axis directly comparable with those of Figure 6.18.

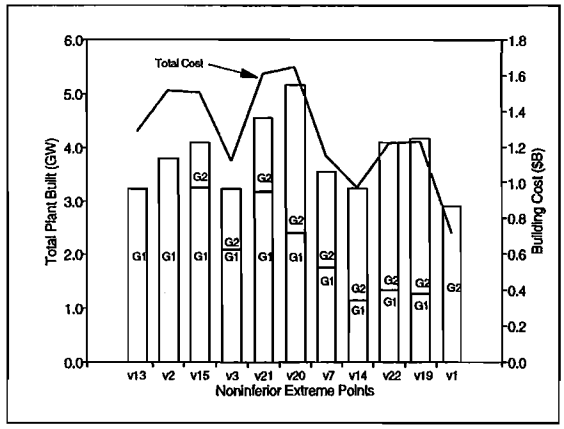

Figure 6.20:
Bar Graph of Building Programmes
This graph shows the total plant capacities, and mixes of plant types, that are specified by the solution at each of the noninferior extreme points.

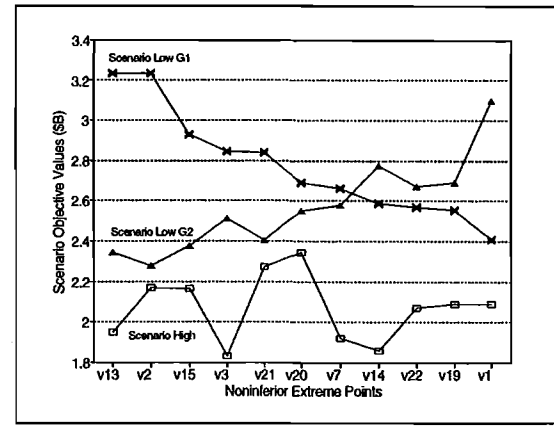

Figure 6.21:
Value Paths of Scenario Objective Values
This graph shows the scenario objective function values for each noninferior extreme point, with the points connected by lines to give a picture of the available trade-offs.

The faces of the noninferior set are represented in Figures 6.18 and 6.19, in which the noninferior set appears as a band of objective function values for each scenario. The bands are quite narrow for scenarios LowG1 and LowG2, and they have quite steep, and opposite slopes. This suggests that trade-offs between these two scenarios will involve greater changes in objective function value than for scenario High. This can also be seen from the fact that it is possible to draw a constant objective value line (for example $z_H = 2$) right across the noninferior set for scenario High, whereas a move from one side of the noninferior set to the other necessitates a change of objective value for both of scenarios LowG1 and LowG2. It is also apparent from Figure 6.22, in which the contour lines for scenario High are close together across the whole of the noninferior set.

Clearly, the decision finally selected will depend on the judgement of the decision maker and her attitudes to risk in this situation. So we complete the example by simulating our decision maker choosing her decision as follows:

She sees from Figure 6.21 that decisions v13, v2 and v15 lead to low costs under scenario LowG2 and high costs under scenario LowG1, while decision v1 leads to low costs under scenario LowG1 and high costs under scenario LowG2. If she chooses one of these decisions she will be gambling on a particular plant type being reliable. If she chooses a solution from the central part of Figure 6.21, then the costs under

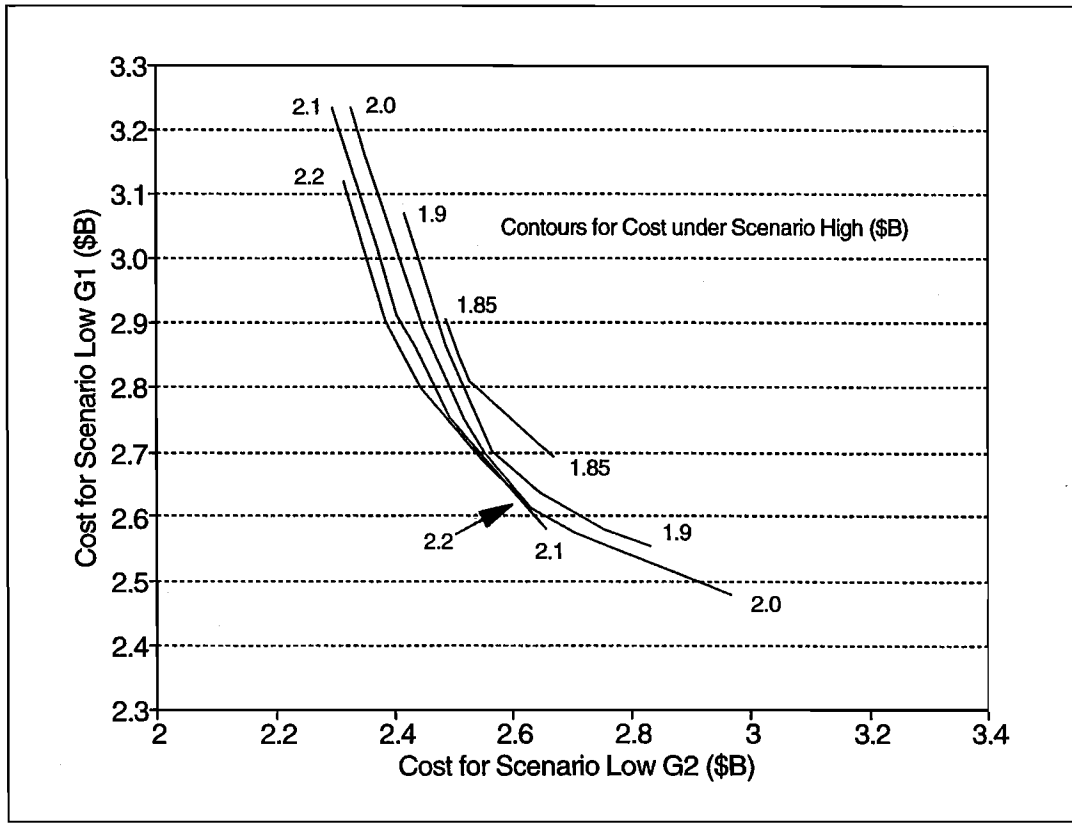

Figure 6.22: Decision Map with Contours for Scenario High
The contours trace out a frontier that is very similar to a two-scenario frontier. The slopes of the contours show that achieving low costs for one of LowG1 and LowG2 imposes a high cost for the other, while approximately equal trade-offs occur when the costs under the two scenarios are approximately equal. The contours are close together, showing that large changes in the cost under scenario High are associated with small changes in cost for the other scenarios.

the two scenarios will be similar. In this central region, from v3 to v19, the costs under LowG1 and LowG2 are in a range from $B2.85 to $B2.4. If she also wants to hold down the costs under the High scenario, then Figure 6.18 shows that there is a region on faces f6, f10 and f39 over which the cost under the High scenario is low and the costs under the other two scenarios are approximately equal. She decides that this is the best region to be in, because it provides reasonable certainty of outcomes under scenarios LowG1 and LowG2, while producing a good result if plant availabilities are high. She could increase the cost under the High scenario to obtain lower costs under the other two scenarios, by moving towards solution v20. However, the ranges of outcomes for scenarios LowG1 and LowG2 are quite narrow, which shows that little will be gained under those scenarios, even when

large increases in cost are imposed on scenario High. This can clearly be seen in the Decision Map of Figure 6.22 from the fact that the contour lines for scenario High are very close together.

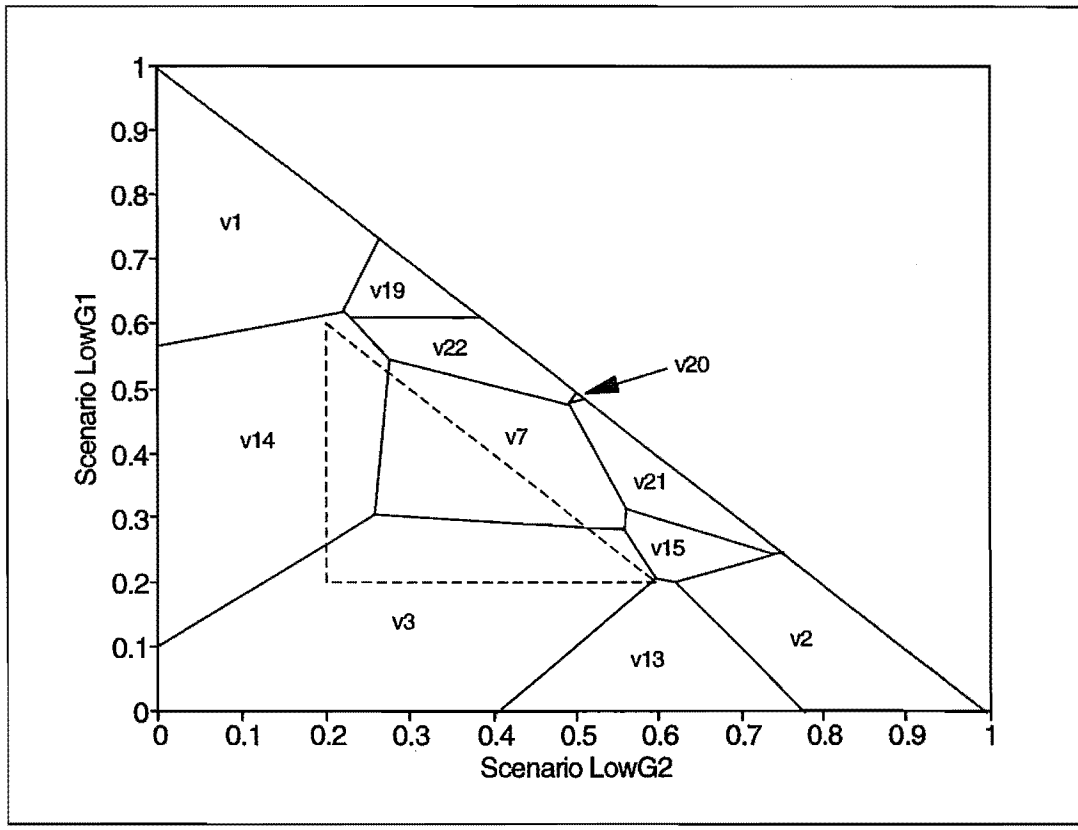

Figure 6.23: Noninferior Points Plotted in Weighting Space
The noninferior extreme points are mapped onto the weight space. The weight placed on scenario High is on the vertical axis. Along the diagonal line, the weight placed on scenario High is zero. The dotted triangle marks the region within which no scenario has a weight of less than 0.2 (or more than 0.6). The optimal solution for any weighting of the scenarios can be read from the figure. For example, v14 is optimal for the weighting vector (0.2, 0.4, 0.4).

From Figure 6.22 she sees that she could choose the point at the bottom end of the 1.85 contour where the costs under the LowG1 and LowG2 scenarios are approximately equal at \$B2.7. From that point she can trade-off against the High scenario, while keeping the costs under the other two scenarios equal, by moving diagonally down and to the left until she reaches the point where the cost under the High scenario is \$B2.2 and under the other two is a bit more than \$B2.6. However, at this point the contours for 2.2, 2.1 and 2.0 almost coincide, so that she may as well set the cost under the High scenario to \$B2.0. In the end she decides that the cost

under the High scenario should be \$B1.9, because the distance between contours 1.85 and 1.90 is quite large, whereas the distance from contour 1.9 to 2.0 is much smaller. This means that the rates of improvement of the outcomes under LowG1 and LowG2 are larger from 1.85 to 1.90 than beyond 1.90, and so the trade-off becomes much less attractive as the cost under the High scenario increases beyond 1.90.

In Figure 6.23 the dotted triangle marks the region within which no scenario is given a weight less that 0.2, or greater than 0.6, and so solutions within the region are appropriate when the scenarios are considered to be of similar importance. This region includes the three solutions v14, v7 and v3, and face f10, which suggests that, under this criteria, it would be appropriate to select the final solution from face f10 (which includes v14, v7 and v3).

As can be seen on Figure 6.18, in the region where the costs under LowG1 and LowG2 are approximately equal, the contour $z_H = 1.9$ crosses faces f1, f6, f10, f39, f40 and f33. By setting the cost under the High scenario to \$B1.9, and requiring the costs for the other two scenarios to be equal, a criterion vector can be calculated for each face. For example, for face f10 the two equations:

$$0.3061 z_{G1} + 0.2580 z_{G2} + 0.4359 \times 1.90 = 2.319 \tag{6.3}$$

$$z_{G1} = z_{G2} \tag{6.4}$$

are solved, to give $z_{G1} = z_{G2} = 2.642$.

Of the five faces, face f10 produces the lowest cost for scenarios LowG1 and LowG2 (\$B2.642), and so the criterion vector, (2.642, 2.642, 1.900), is chosen as the preferred criterion vector.

The corresponding decision vectors can be found from the geometry of the problem. First the convex combination operator, γ, that calculates the criterion vector from the corner points of face f10, is found by solving:

$$[\mathbf{v14} \,|\, \mathbf{v7} \,|\, \mathbf{v3}]\gamma = (2.642, 2.642, 1.900)^T \tag{6.5}$$

which gives $\gamma^T = (0.328, 0.645, 0.026)$

The solution vector for the preferred criterion vector is then found by solving

the equations:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{G1} \\ \mathbf{y}_{G2} \\ \mathbf{y}_{H} \end{bmatrix} = \begin{array}{c} \begin{matrix} \text{v14} & \text{v7} & \text{v3} \end{matrix} \\ \begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{y}_{G1} & \mathbf{y}_{G1} & \mathbf{y}_{G1} \\ \mathbf{y}_{G2} & \mathbf{y}_{G2} & \mathbf{y}_{G2} \\ \mathbf{y}_{H} & \mathbf{y}_{H} & \mathbf{y}_{H} \end{bmatrix} \end{array} \gamma$$

where: $\mathbf{x}$ = stage one vector for each extreme point

$\mathbf{y}_{G1}$ = recourse vector under scenario LowG1 for each extreme point

$\mathbf{y}_{G2}$ = recourse vector under scenario LowG2 for each extreme point

$\mathbf{y}_{H}$ = recourse vector under scenario High for each extreme point

This produces the solution shown in Table 6.21.

Scenario (cost)		LowG1 (2.642) Despatch			LowG2 (2.642) Despatch			High (1.900) Despatch		
Plant	Built	Base	Mid	Peak	Base	Mid	Peak	Base	Mid	Peak
G1	1.563	0	0.776	0.006	0.001	1.051	0.354	1.000	0.406	0
G2	1.864	0	0.397	1.094	0	0	0.746	0	0.494	1.000
Buy	—	1.100	0.028	0	1.099	0.149	0	0	0	0
Totals:	3.427	1.100	1.201	1.100	1.100	1.200	1.100	1.000	0.900	1.000
Cost:	1.091	1.100	0.342	0.110	1.099	0.359	0.092	0.430	0.279	0.100

Table 6.21: Calculated Solution for the Preferred Criterion Vector
This solution builds 3.427 GW of plant, and buys Base and Middle load power under the LowG1 and LowG2 scenarios. No power is bought under the High scenario.

Because the face is not known to be noninferior, a better criterion vector may be available. To obtain this vector, problem (6.3) can be solved once more with the scenario weights, $p^{\omega} = 1$ and upper bounds on the scenario objectives of \$B2.642 for scenarios LowG1 and LowG2, and \$B1.9 for the High scenario. This produces the solution shown in Table 6.22, which is not significantly different from the solution obtained using the geometry of the problem. Clearly, there is no point in finding the preferred solution by geometry if the decision maker is going to solve problem (6.3) to find a truly noninferior solution. The approach chosen will depend on the maximum allowable error used when the approximation was found, and on

the size of problem (6.3). Of course, if the decision maker's preferred criterion vector lies on a face that is known to be noninferior, then nothing will be gained by solving problem (6.3), and the preferred solution should be found using geometry.

Objective Values ($B)			Plant Sizes (GW)		Plant
LowG1	LowG2	High	G1	G2	Cost
2.642	2.641	1.900	1.567	1.861	1.092

Table 6.22: The Final Preferred Solution
The coordinates in criterion space of the final preferred solution, and the plant sizes, and construction costs of the stage one decision.

6.3.1 Evaluation of the Decision

The issue now arises of how to evaluate the quality of this decision. The value of the stochastic solution (VSS) is used in the literature (e.g. Birge 1995) as a measure of the quality of stage one decisions. In order to calculate the VSS, the expected value solution is found by solving the problem for the expected value scenario. For problem APL1P, the expected value scenario has the following values for the uncertain parameters. The expectations of the plant availabilities are 0.68 for G1, 0.64 for G2, and of the demands are 1040 MW for all modes. The solution to this problem builds 1529 MW of G1 and 1625 MW of G2. The whole base load (1040 MW) is bought from outside, and G1 supplies the mid load (0.68×1529) and G2 the peak load (0.64×1625). The cost of the expected value solution is ($1529 \times 4 + 1625 \times 2.5 + 1040 \times (10 + 2 + 1) = \$B2.370$). However, the expected cost of this solution, over all 1280 scenarios, is \$B2.470. The stochastic optimal solution has an expected cost of \$B2.464, and so the VSS is \$B0.006 (0.24%), which is approximately zero.

The stochastic optimal solution, the expected value solution, and the NSSA solution are all listed in Table 6.23. These stage one decisions were each tested under the 1280 scenarios of APL1P. This was done by fixing the stage one decision at the value of the solution being tested, and then solving the stage two problem for every scenario. The problem was also solved for each scenario individually, generating 1280 scenario-optimal stage one decisions. This is referred to in the literature as the "Wait and See" formulation (for example, see Birge 1995), and the expectation

of the 1280 scenario-optimal stage one solutions is the expected value with perfect information.

		NSSA Solution	Stochastic Optimal Solution	Expected Value Deterministic Solution	Individual Scenario Solutions
Plant (G1, G2)	(MW)	(1567,1861)	(1800, 1571)	(1529, 1625)	
Building Cost	($\$10^9$)	1.092	1.113	1.018	
Expected Cost	($\$10^9$)	2.467	2.464	2.470	2.190
Average Cost	($\$10^9$)	2.705	2.695	2.696	2.270
Variance	($\$10^{18}$)	41	41	39	19.7
Minimum Cost	($\$10^9$)	1.796	1.770	1.729	1.602
Maximum Cost	($\$10^9$)	4.543	4.542	4.473	3.600

Table 6.23: Model APL1P: Comparison of Solutions
The expected cost is the sum over the scenarios of the cost of the solution multiplied by the probabilities of the scenarios. The average cost is the mean across all scenarios without reference to their probabilities. The minimum and maximum costs are the best and worst outcomes across all scenarios.

It is clear that, for this problem, there is nothing to be gained by solving the stochastic optimisation formulation instead of the expected value formulation. Although the value of perfect information is \$B0.274, the Value of the Stochastic Solution is approximately zero. The NSSA solution chosen by this decision maker is also no better (or worse) than the expected value deterministic solution. However, the process of carrying out the analysis and choosing a decision to implement has given the decision maker greater insight into the problem than can be obtained using either the stochastic optimisation formulation or the expected value formulation. The strength of Noninferior Set Scenario Analysis does not lie in its ability to find good expected value solutions, but in its ability to produce a great deal more information than is available from the single solution produced by stochastic optimisation, and so provide greater insight into the problem. It also has less demanding data requirements than both the stochastic optimisation formulation and the expected value formulation, because probabilities do not have to be assigned to the scenarios.

A very important insight that NSSA makes available is that the surface of the weighted objective function is quite flat. This can be seen from the "Weighted Value" column of Table 6.20 (page 186), which shows that the range of the weighted

objective function values across the noninferior faces is from $B2.319 to $B2.618, or about 12%.

The optimal expected value solution to this problem is quite insensitive to the weights placed on the scenarios, provided that no scenario is given an extreme weight. In Figure 6.23 (page 190), solutions v14, v7 and v3 are optimal inside the dotted triangle, within which the minimum weight placed on any scenario is 0.2. From Figure 6.20 it can be seen that the stage one decisions for these three solutions are quite similar. The total capacity built is around 3.2 GW, and the proportions of the two types of plant built changes from 2:1 to 1:2. This, combined with the fact that these three solutions are the vertices of the same noninferior face (f10), shows the decision maker that she can adjust the sizes of the two plants over quite a wide range without greatly altering the outcomes under the scenarios.

Neither the solution of the expected value deterministic problem, nor of the full stochastic formulation make these insights available because only one point on the objective function surface is produced. This information could only be obtained for these formulations by expending additional effort on sensitivity analysis.

The result of solving the stochastic optimisation problem is the single recommendation that 1.80 MW of plant G1 and 1.57 GW of plant G2 be built, and that the expected cost of this decision is $B2.464. On the other hand, the output from Noninferior Set Scenario Analysis provides eleven noninferior solutions. The decision maker can choose one of these solutions, or use the approximation to the noninferior set to identify additional noninferior solutions. The approximation to the noninferior set provides insights about the problem, and quantifies the trade-offs that are available to the decision maker as she looks for a decision that will prepare adequately for the uncertain future.

In this example the uncertainty has been summarised as 3 scenarios, rather than 1280, which means that the decision maker can look at the scenarios and consider what they mean, and how different decisions will perform under each of them. Although this reduction to three scenarios means that much detail has been lost, it is very unlikely that any decision maker would be capable of considering 1280 scenarios anyway.

Stochastic optimisation presents the decision maker with a single solution that she has to accept, reject, or attempt to modify in light of non-quantifiable aspects of the problem. Noninferior Set Scenario Analysis presents the decision maker with

a set of alternatives to choose from, and non-quantifiable aspects of the problem can readily be included in this selection process. For example, suppose that the suggestion that plant G1 might be built has lead to vigourous opposition from the local community, and the decision maker is faced with the prospect of protracted wrangling, and possible court action. However, it appears that a reduction in the size of the plant would moderate the opposition.

The decision maker can use the noninferior set to look for alternatives that might reduce opposition to the expansion of generating capacity. Solution v1 builds no G1 plant at all, but the cost under scenario LowG2 is unacceptably high. Of the remaining noninferior solutions, v14 builds the smallest G1 plant, being 1.11 GW, rather than the 1.57 GW built by the preferred solution. The cost for scenario LowG2 under this solution is \$B2.77, \$B0.14 more than under the preferred solution. However, there are offsetting reductions in the costs under the other two scenarios (\$B0.058 for LowG1, and \$B0.038 for High). Solutions v19 and v20 also build small G1 plants, and these solutions are the vertices of faces f33 and f40. The tradeoffs that are available from moving around on these faces can be seen on Figures 6.18 and 6.19 (page 187). Similarly, the tradeoffs involved in increasing the size of plant G1 and moving towards the preferred solution are given by faces f39, f10 and f2.

This provides the decision maker with information to use in negotiations with the opposition groups to see if they will soften their position from outright opposition, to a demand that the plant be small. If she had only the stochastic optimum available (G1 = 1.80, G2 =1.57), she would have little information to assist in looking for alternatives to use in her negotiations with the opposition groups.

6.3.2 Computational Effort

When all combinations of the outcomes of all of the uncertain events are generated, problem APL1P consists of 1280 scenarios, and problem (6.3) has 7,680 constraints and 12,802 variables. When this problem was solved exactly, using AMPL and CPLEX on a Pentium 166, AMPL took 187 seconds to build the model, and CPLEX took 11,227 simplex iterations, and 89 seconds to solve it, giving a total elapsed time of 274 seconds.

When three scenarios are used to analyse problem ALP1P using Noninferior Set

Scenario Analysis, problem (6.3) has 32 variables and 18 constraints. The characterisation of the noninferior set used above was found in a total elapsed time of 47 seconds. CPLEX was called 26 times to solve problem (6.3) using a total of 197 iterations, and it was called 84 times to solve problem (5.4), which determines the maximum possible error for each face in the approximations. These 84 calls used a total of 645 iterations. The final stage, in which the noninferior faces are identified and checked (refer to Sections 5.3.5 and 5.3.7) called CPLEX 6 times and used 92 iterations.

Clearly, Noninferior Set Scenario Analysis required significantly less effort to find a set of noninferior solutions than was required to solve the full stochastic formulation exactly. When total elapsed times are compared, Noninferior Set Scenario Analysis used 17.2% of the time required to solve the full stochastic formulation. Comparison of the CPLEX iterations shows that NSSA used 8.3% of the iterations required to solve the full problem.

Importance sampling reduces the computational effort required to solve the problem by sampling from the set of 1280 scenarios. Infanger (1994) reports that "the method achieves with about 2.9% of the computation effort a solution that is with 95% confidence within an interval of $\pm$ 2.1% of the correct answer.". This certainly suggests that importance sampling is an effective response to the computational intractability of stochastic optimisation problems. However, the fundamental shortcomings of stochastic optimisation remain. Stochastic optimisation assumes that probabilities are available for the scenarios, and that the decision maker is sufficiently risk neutral to be willing to accept the single solution produced.

In contrast, Noninferior Set Scenario Analysis does not assume that probabilities are available for the scenarios, and it produces a set of noninferior solutions for the decision maker to choose among. This set of alternative solutions, and their performances under the different scenarios, provides the decision maker with much greater insight into the problem than can be obtained from a single "optimal" solution.

6.4 The Choice of Scenarios

An issue which is of critical importance to both stochastic optimisation and scenario analysis is the choice of scenarios. In scenario analysis the scenarios are created explicitly to describe ways in which the future may evolve. In stochastic optimisation

they may also be created explicitly, or the analyst may use random variables to model the uncertainty and have these random variables implicitly generate the scenarios.

In this section we will derive other scenarios from the data of problem APL1P, and use them to analyse the problem. The objective of this section is to gain some insights into how the choice of scenarios influences the set of alternatives that are identified. The first set of scenarios consists of three extreme scenarios, an optimistic scenario, a pessimistic scenario, and a middle one. The second set consists of three less extreme scenarios. Neither set of scenarios performs very well, the first because the scenarios are too extreme, and the second because the scenarios do not combine conflicting outcomes of the uncertainties. In both cases the scenarios fail to identify contrasting alternatives to choose among, and they also fail to bring out the trade-offs involved in selecting a decision to implement.

6.4.1 Extreme Scenarios

In this section the scenarios will be chosen to represent a good outcome, a bad outcome, and a middle outcome. The values used for these scenarios are shown in Table 6.24. Scenario Bad is highly pessimistic. It pitches very low availabilities against the highest possible demand. Scenario Good, on the other hand, assumes that both plants are fully available, and that demand is very low. Scenario Exp is the expected value scenario. This set of scenarios consists of a middle scenario, contrasted by two extreme scenarios. The results of analysing the problem, as represented by these scenarios, are summarised in Tables 6.25 to 6.26, and Figures 6.24 to 6.27.

	Availabilities		Demands		
	G1	G2	Base	Mid	Peak
Scenario Bad	0.1	0.1	1200	1200	1200
Scenario Exp	0.68	0.64	1040	1040	1040
Scenario Good	1.0	1.0	900	900	900

Table 6.24: Model APL1P: Extreme Scenarios
Scenario 'Bad' pitches very low plant availability against very high demand, while scenario 'Good' pitches very high plant availability against very low demand, and scenario 'Exp' used expected values.

The faces and edges are listed in Table 6.26, and the approximation to the noninferior set is shown in Figure 6.25. Because the noninferior set includes isolated

Solutions	Objective Values ($B)			Plant Sizes (GW)		Plant
	Bad	Exp	Good	G1	G2	Cost
v2	4.327	2.370	1.729	1.529	1.625	1.018
v3	4.293	2.498	1.602	1.800	.900	0.945
v12	4.163	2.437	1.647	0.900	1.800	0.810
v16	4.120	2.373	2.046	0.0	3.250	0.813
v4	3.888	2.567	1.800	0.0	1.800	0.450
v21	3.860	2.590	1.861	0.0	1.625	0.406
v15	3.744	2.827	2.115	0.0	0.900	0.225
v1	3.600	3.120	2.700	0.0	0.0	0.0

Table 6.25:
The Noninferior Extreme Points for the Extreme Scenarios
The coordinates in criterion space of the noninferior extreme points, and the plant sizes built by the initial decisions. The points are sorted in descending order by their values under scenario Bad.

Faces	Vertices			Neighbouring Faces			Scenario Weights Bad	Exp	Good	Weighted Value
f19	v16	v12	v4	f6	f21	f28	0.3279	0.5481	0.1240	2.905
f21	v16	v12	v2	f11	f19	f20	0.2068	0.6634	0.1298	2.691
e1	v1	v15		f33			0.6708	0.3293	0.0	3.442
				f7			0.8025	0.0	0.1975	3.422
e2	v4	v21		f27			0.6708	0.3292	0.0	3.442
				f26			0.6863	0.0	0.3137	3.233
e3	v15	v21		f19			0.3279	0.5481	0.1240	2.905
				f16			0.3579	0.0	0.6421	2.547
e9	v3	v12		f9			0.0	0.4261	0.5739	1.984
				f3			0.2564	0.0	0.7436	2.292

Table 6.26: Faces in the Approximation to the Noninferior Set
The approximation to the noninferior set is made up of two faces, and four edges. Each face is shown with its vertices, its neighbouring faces, and the weighting vector for which it is the optimal solution to the weighting problem. For each edge, the vertices at its ends are shown, along with the face on each side of the edge. Every point on an edge is an optimal solution to the weighting problem for any convex combination of the weights of the faces on each side.

edges, the decision map will include discrete points, as in Figure 6.26. The contour lines relate to the two faces, f19 and f21. The discrete points, labelled with the objective value under scenario Good, occur where the contour lines cut through the noninferior edges. The point, 1.6, is on edge e9, while the other points are on the edges e2, e3, and e1.

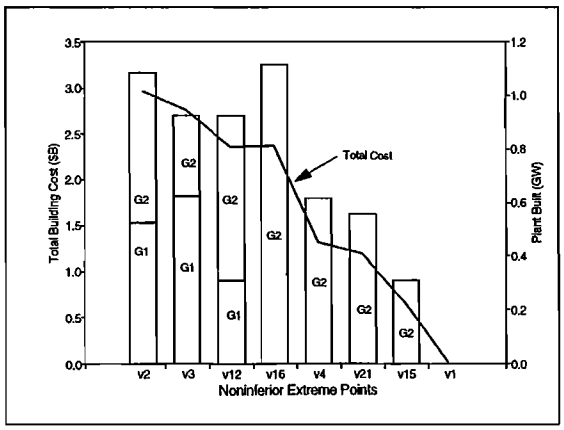

Figure 6.24:
Building Programmes for the Extreme Scenarios
This graph shows the total plant capacities, and mixes of plant types, that are specified by the solution at each of the noninferior extreme points.

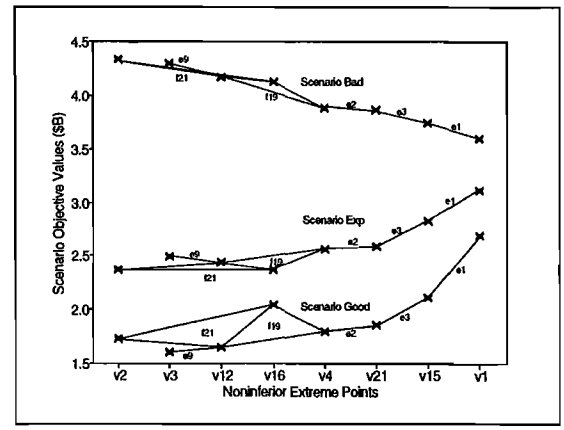

Figure 6.25:
The Noninferior Set for the Extreme Scenarios
The noninferior set displayed for each scenario individually. This set consists of two faces and four edges.

The first observation that can be made from this output is that the best protection against very low plant availabilities is to build no plant at all. However, this is also apparent from Table 6.17 (page 184), which shows that at plant availabilities of 0.1, the cost of building plant to supply power is 2 to 3 times the cost of buying it. The second observation is that the scenarios Exp and Good have very similar slopes across the noninferior set. The principal trade-off is between the scenario Bad and scenarios Exp and Good. This trade-off is particularly apparent as the plant capacity built is reduced towards zero (to the right of solution v4 in Figure 6.25). There is a reduction in cost for scenario Bad and an increase in cost for both of scenarios Exp and Good.

These scenarios do not produce particularly useful results because scenarios Exp and Good are rather similar, in that the plant availabilities are matched to the demands, while scenario Bad models an outcome that is so extreme that very little can be done about it. Another shortcoming of the two extreme scenarios is that all

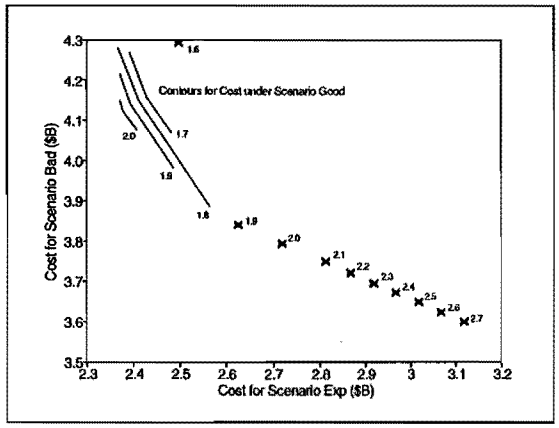

Figure 6.26:
Decision Map with Contours Shown for Scenario Good
The Decision Map consists of continuous contours over the faces of the noninferior set. Where the noninferior set is made up of edges, the Decision Map consists of discrete points.

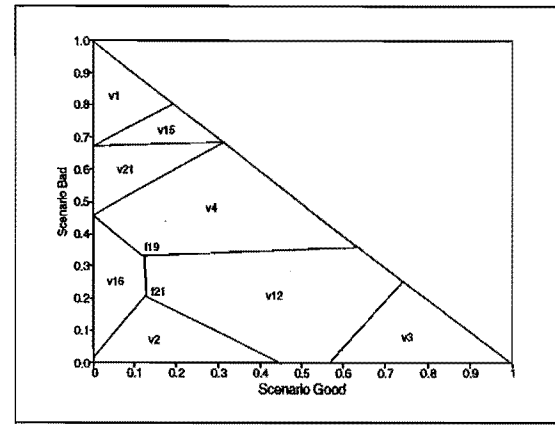

Figure 6.27:
Scenario Weights for the Extreme Scenarios
The Noninferior Set in weighting space.

other possibilities lie to one side of each of them. That is, for the Bad scenario, all demands are lower than the demands in the scenario, and all plant performances are better than the ones in the scenario. Similarly, for the Good scenario, all demands are higher than the ones in the scenario, and all plant availabilities are lower than the ones in the scenario. Because of this, these scenarios represent only a small sample of the possible futures, whereas scenarios that are less extreme, such as those used in Section 6.3, represent a wider range of possible futures, and so model the uncertainty more fully.

Under each of the scenarios the two plant types have similar availabilities which means that outcomes under which one of the plants performs well, while the other performs poorly, are not represented at all. This means that the effects of changing the mix of plant, so that each plant can provide a hedge against low availability of the other, are not modelled.

The only trade-offs that are identified, are between building plant in the expectation that performance will be satisfactory, and buying power to meet all demand in the expectation that plant performance will be extremely bad. It can be seen that these scenarios provide few insights about the problem that could not be obtained by studying the data. What is required are scenarios that are both less extreme, and

that combine contrasting outcomes for the uncertainties. These scenarios would be improved by bringing contrasting outcomes together in a scenario (e.g. by matching high availability for one plant with low availability for the other), and by positioning them to represent a wider range of futures. In particular, by choosing them to be within the range of possible futures, rather than at the boundaries.

6.4.2 Three More Scenarios

The values used for this set of scenarios are shown in Table 6.27. Again scenario 1 is rather pessimistic. It pitches moderate availabilities against the highest possible demand. Scenario 2 is also pessimistic, and pitches lower availabilities against somewhat lower demand. Scenario 3 assumes high availabilities for both plants and moderately high demand. The results of analysing the problem as represented by these scenarios are shown in Tables 6.28 and 6.29, and Figure 6.28.

	Availabilities		Demands		
	G1	G2	Base	Mid	Peak
Scenario 1	0.71	0.68	1200	1200	1200
Scenario 2	0.51	0.70	900	1200	1200
Scenario 3	0.90	0.90	1100	1100	1100

Table 6.27: Model APL1P: Three More Scenarios

Solutions	Objective Values ($B)			Plant Sizes (GW)		Plant
	S1	S2	S3	G1	G2	Cost
v3	2.730	2.764	2.086	2.444	1.222	1.283
v1	2.669	2.742	2.554	3.380	1.765	1.793
v16	2.681	2.633	2.101	1.952	1.714	1.209
v4	2.676	2.623	2.103	1.902	1.765	1.202
v12	2.678	2.515	2.123	1.222	2.444	1.100
v14	2.679	2.459	2.208	0.877	2.789	1.048
v17	2.682	2.368	2.381	0.097	3.429	0.996
v2	2.698	2.357	2.392	0.0	3.429	0.857

Table 6.28: Noninferior Extreme Points, Three More Scenarios
The coordinates in criterion space of the noninferior extreme points, and the plant sizes built by the initial decisions. The points are sorted in descending order by their values under scenario 1.

Faces	Vertices			Neighbouring Faces			Scenario Weights S1	S2	S3	Weighted Value
f23	v17	v14	v2	f6	f4	f30	0.1647	0.5491	0.2862	2.423
f19	v12	v4	v1	f24	f3	f9	0.9700	0.0204	0.0097	2.670
f24	v14	v12	v1	f19	f30	f8	0.9667	0.0274	0.0059	2.671
f30	v14	v17	v1	f24	f23	f29	0.9659	0.0320	0.0021	2.671
e4	v3	v16		f28			0.0	0.1010	0.8990	2.155
				f10			0.2327	0.0	0.7673	2.236
e5	v4	v16		f27			0.0	0.1563	0.8438	2.184
				f10			0.2327	0.0	0.7673	2.236

Table 6.29: Noninferior Faces, Three More Scenarios
The approximation to the noninferior set is made up of four faces and two edges. All points on a face are optimal solutions for the weighting vector corresponding to the outward normal of the face. All points on Edge e8 are optimal solutions for all convex combinations of the weighting vectors of the faces on each side of it (f2 and f3). Although they are not noninferior, these faces are included in the table to complete the description of Edge e8.

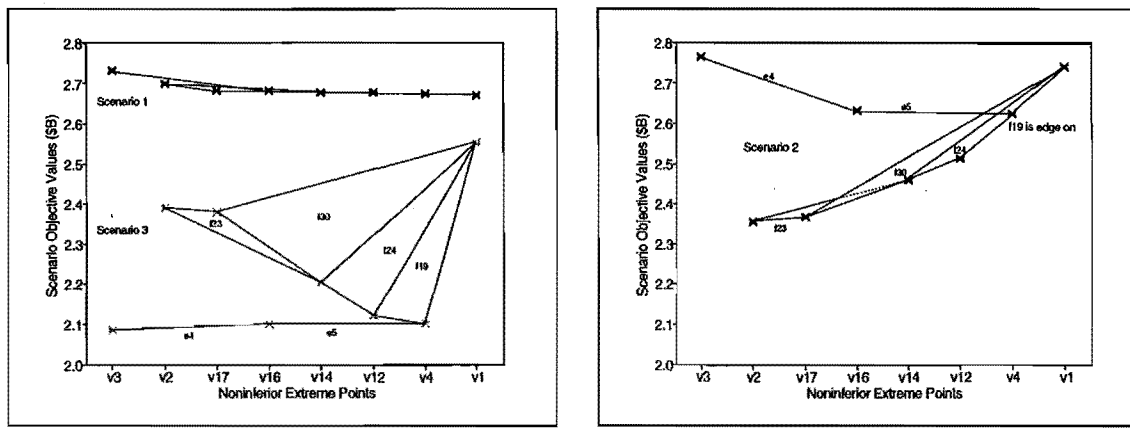

Figure 6.28: The Noninferior Set for Three More Scenarios
The noninferior set displayed for each scenario individually. This set consists of two faces and four edges.

Scenario 1 is not very helpful, both because it is very extreme, and because the plants have the same availabilities which means that it cannot provide information about trade-offs between the two plant types. Its scenario-maximal solution, v1, trades off performance under scenario 1 against both scenarios 2 and 3, but the trade-offs involved in moving between the remaining solutions are principally between scenarios 2 and 3. From Table 6.29 it can be seen that the normals of the three faces f19, f24 and f30 are almost parallel to the scenario 2–scenario 3 plane in criterion

space, which is why the trajectory of scenario 1 is almost horizontal over that region of the noninferior set.

The trade-offs between scenarios 2 and 3 can be presented in two dimensions, as shown in Figure 6.29. The two dimensional trade-off frontier is shown as a solid line, and the faces are shown with dotted lines. Solution v1 is inferior in two dimensions, because it trades off both scenario 2 and scenario 3 to improve performance under scenario 1. The remaining solutions are noninferior in this two dimensional space. There is an abrupt change in the slope of the frontier at v12, which will draw the decision maker to this solution, because to move away from it trades a large reduction in performance under one scenario for a small improvement under the other. However, solution v12 builds much more of plant G2 than of G1, despite the fact that G1 is, in fact, the more reliable plant. From Figure 6.30 it can be seen that the proportion of plant G1 increases as the solution moves from v12 to v4, and the decision maker would be likely to choose a decision close to v4 in order to increase the proportion of the more reliable plant in the installed capacity.

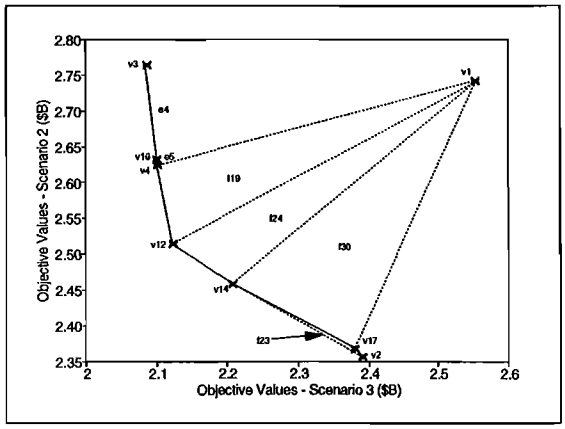

Figure 6.29:
Two Dimensional Trade-off Frontier
The noninferior set displayed as a two-dimensional frontier for scenarios 2 and 3. The noninferior faces are shown dotted. The criterion vectors on the solid line are noninferior in two dimensions. Criterion vector v1 is dominated in two dimensions, but nondominated in three dimensions.

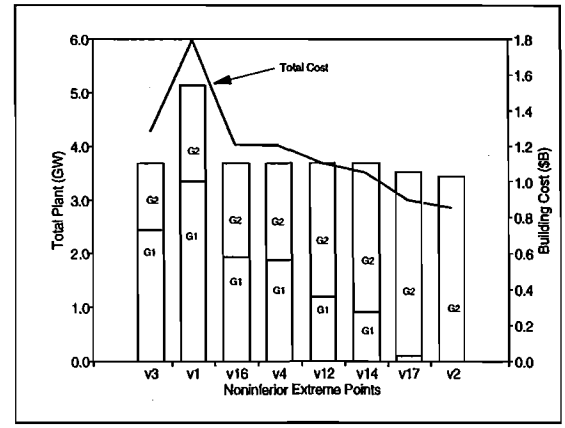

Figure 6.30:
Building Programmes for Three More Scenarios
This graph shows the total plant capacities, and mixes of plant types, that are specified by the solution at each of the noninferior extreme points.

The scenarios chosen for this example also turn out to be a poor choice. Scenario 1 provides almost no information, and the final decision can be chosen without reference to it. However, scenarios 2 and 3 are also inadequate, because they do not

model the relative reliabilities of the two plant types correctly. They represent plant G1 as being less reliable than plant G2, when in fact, the opposite is true. Consideration of the relative reliabilities of the plant types could be expected to lead the decision maker to select a final decision that is different from v12, the decision that clearly appears from the trade-off frontier to be the best one. This strongly suggests that these scenarios have failed to adequately represent this important aspect of the problem.

Nevertheless, this example does illustrate the strength of the NSSA approach, in that the decision maker has other decisions to choose among if she decides that the "best" solution is unacceptable. It also provides a framework for choosing among these solutions.

6.4.3 Summary

These trials suggest that the choice of scenarios affects the insights that can be gained from the analysis, and that the scenarios should be chosen to provide a balanced picture of the uncertainties. If the scenarios are very similar, they will produce similar decisions and fail to identify the full range of alternatives available to the decision maker. If they are too extreme, the recourse decisions can be very tightly constrained with few adjustments available to the decision maker. This results in the scenario objective function values being restricted to narrow ranges, and few of the trade-offs between the available decisions will be apparent.

The scenarios should be designed to accentuate the issues. In the example in Section 6.4.2, scenario 2 fails to do this because the low availability of plant G1 is offset by the low demand for base load, which is the load that plant G2 cannot supply economically. This scenario would have been more useful if the low availability for plant G1 had been matched with a high demand for base load, and the high availability of plant G2 matched with reduced demand for mid or peak load. This combination would have captured the fact that G2 is not an economically attractive replacement for plant G1 when base load has to be supplied. Similarly, the Exp scenario in Section 6.4.1 sets the availabilities for both plants below the level at which they can meet base load more cheaply than buying power, and both of them above the level at which they are the cheapest option for supplying mid and peak load.

The scenarios must be designed to accentuate the issues that are important to the decision maker, and to bring out the trade-offs that will be involved in choosing between alternative courses of action.

6.5 Conclusions

In this chapter we have used two examples from the literature to demonstrate the application of NSSA and to bring out the advantages this approach offers over the standard stochastic optimisation formulation. Firstly, NSSA dispenses with the need to assign probabilities to the scenarios, which can be very difficult to do when modelling situations that have seldom, if ever, occurred before. In stochastic optimisation, when there is uncertainty about the scenario probabilities, sensitivity analysis should be carried out on the probabilities. Unfortunately, when there are more than two scenarios this is a difficult and computationally demanding exercise. In contrast, the noninferior set produced by NSSA can be interpreted to provide a complete picture of how the optimal expected value solution will change as the scenario probabilities change.

NSSA produces a set of nondominated solutions that provide a great deal more information than is available from the single optimal solution provided by stochastic optimisation. This is of particular value when the object of the modelling is to provide insights to support a decision process, rather than to provide the answer. When modelling strategic problems this will normally be the case because there will be important, non-quantifiable issues that cannot be represented in the mathematical programming model.

Models built to find the noninferior set can avoid the need to include arbitrary constraints in a way that is not possible with stochastic optimisation. The limited availability of funds in the Smeers and Louveaux example, and the minimum plant size in the APL1P example, are cases in point. Frequently the exact values of the RHS's of such constraints are unclear, but values must be chosen when using stochastic optimisation. The question of the sensitivity of the optimal solution to changes in these values then arises. When using NSSA the constraint can simply be converted to the calculation of the quantity of the resource required. However, it may be desirable to set extreme upper or lower bounds on the use of the resource to avoid reporting solutions that are quite impractical. The resource requirement is

reported to the decision maker as part of the solution, and can be included in the evaluation of the alternatives.

Although NSSA is computationally demanding, this can be controlled to some extent by using an iterative approach in which a coarse approximation to the non-inferior set is found first. Regions of interest are identified and the approximation refined over those regions only. The solution of a stochastic optimisation model may be less computationally demanding than an analysis using NSSA, but it will not produce as much information. In fact, the decision maker may well request additional runs to gain further insights, insights that would be readily available from an NSSA analysis.

In the Smeers and Louveaux example, the computational effort required to carry out the NSSA analysis was much greater than that required to find a single optimal solution. However, for the APL1P example, by using three scenarios instead of 1280, NSSA was able to find good solutions with considerably less effort than was used by stochastic optimisation to find a single solution.

Chapter 7

Mixed Integer Problems

7.1 Introduction

In many strategic planning problems, there are courses of action that must be modelled using binary variables. For example, a dam can be built this period or next period, but not both; if a factory is built, it must be at least some minimum size; if the capacity of a power station is increased, its transmission line must upgraded. Typically, the model also includes continuous variables, so that it becomes a mixed integer problem.

Scenario planning problems present some special difficulties when they include binary variables because the noninferior set becomes discontinuous, and nonconvex. Some of the noninferior points in criterion space are dominated by (infeasible) convex combinations of other noninferior points, which means that they cannot be found using weighted-sums methods. This has important implications for stochastic programming, because the objective function of a stochastic program is, in fact, the weighted sum of the scenario objectives. (See problem (4.7) in Section 4.4.) This means that the optimal solution to a mixed integer stochastic program may not be the solution that best matches the scenario probabilities used to form the weighted objective.

In this chapter we develop a multi-criteria branch and bound procedure for analysing this problem. At each node in the branch and bound tree the subproblem is solved for each scenario to find a vector of scenario objective function values. The node can be fathomed if the vector at the node is dominated by the vector of scenario objective function values of a known integer solution. Unlike the conventional

branch and bound algorithm, this algorithm cannot use the best solution found so far as a single incumbent value with which to fathom nodes in the tree. This is because no solution can be said to be "best", it can only be said to be nondominated, which means that the "incumbent" is a list of criterion vectors. It also means that new nondominated criterion vectors must be added to the list, rather than replacing the current incumbent. Because the objective is to find a set of nondominated solutions, nodes can only be fathomed if it can be shown that all integer solutions further down the tree will be dominated by a known integer solution. From a computational point of view there are two major difficulties. The first is that the subproblem at each node must be solved for each scenario, rather than once as in single objective branch and bound. The second is that checking whether a criterion vector is dominated does not fathom a node as strongly as checking to see if a single objective function value is less than an incumbent value.

In Section 7.2 we describe the structure of the problem that is to be analysed. In Section 7.3 we briefly discuss the issue of unsupported, nondominated criterion vectors in the context of this problem, and the implications for stochastic programming with integer variables. In Section 7.4 we develop the fathoming rules required for the multicriteria branch and bound algorithm, and we describe this branch and bound procedure in Section 7.5. In Section 7.6 we present the algorithm, and we summarise the chapter in Section 7.7.

7.2 Problem Structure

We assume that the problem is structured so that the stage one variables are all binary, and the stage two (recourse) variables are all continuous. That is, the problem consists of a set of choices at stage one, each of which can be taken, or not taken, and a set of operating decisions at stage two. This problem structure approximates the structure of many strategic planning problems in which investment decisions taken now will determine the organisation's operating parameters in the future. This structure also makes the problem more computationally tractable than problems in which both stages include integer and continuous variables. The problem has the form:

$$\text{maximise} \quad \left\{ \begin{array}{rl} z_1 = & \mathbf{c}_0\mathbf{x} + \mathbf{c}_1\mathbf{x} + \mathbf{q}_1\mathbf{y}_1 \\ z_2 = & \mathbf{c}_0\mathbf{x} + \mathbf{c}_2\mathbf{x} + \mathbf{q}_2\mathbf{y}_2 \\ \vdots & \vdots \\ z_S = & \mathbf{c}_0\mathbf{x} + \mathbf{c}_S\mathbf{x} + \mathbf{q}_S\mathbf{y}_S \end{array} \right\} \tag{7.1 a}$$

$$\text{subject to} \quad \begin{aligned} A\mathbf{x} &= \mathbf{b}_0 \\ B_1\mathbf{y}_1 + T_1\mathbf{x} &= \mathbf{b}_1 \\ B_2\mathbf{y}_2 + T_2\mathbf{x} &= \mathbf{b}_2 \\ \vdots \qquad & \vdots \\ B_S\mathbf{y}_S + T_S\mathbf{x} &= \mathbf{b}_S \\ \mathbf{y}_\omega &\geq 0 \\ \mathbf{x} &\in \{0,1\} \end{aligned} \tag{7.1 b}$$

where: z_ω = the objective under scenario $\omega \in \Omega$

$\mathbf{x}$ = the vector of binary decisions at stage one

$\mathbf{c}_0$ = the vector of stage one costs and benefits of the stage one decision vector

$\mathbf{c}_\omega$ = the vector of stage two costs and benefits of the stage one decision under scenario ω

$\mathbf{y}_\omega$ = the vector of recourse decisions at stage two under scenario ω

$\mathbf{q}_\omega$ = the vector of costs and benefits of the stage two decisions under scenario ω

B_ω = the matrix of constraint coefficients at stage two under scenario ω

$\mathbf{b}_\omega$ = the right hand side vector for stage two under scenario ω

T_ω = the matrix of interactions of $\mathbf{x}$ with the stage two decisions under scenario ω

Once values have been chosen for the stage one decision vector, $\mathbf{x}$, the problem decomposes into S stage two subproblems, one for each scenario. These subproblems

are LP's as in problem (7.2):

$$\begin{aligned} \text{maximise} \quad & z_\omega(\mathbf{x}) = \mathbf{q}_\omega \mathbf{y}_\omega + (\mathbf{c}_0 + \mathbf{c}_\omega)\mathbf{x} \\ \text{subject to} \quad & B_\omega \mathbf{y}_\omega = \mathbf{b}_\omega - T_\omega \mathbf{x} \\ & \mathbf{y}_\omega \geq 0 \end{aligned} \tag{7.2}$$

The stage one decision vector $\mathbf{x}$ is moved to the right hand side, because it is now fixed. The S subproblems can be solved to find the criterion vector associated with the stage one decision vector $\mathbf{x}$. That is, the criterion vector $\mathbf{z}(\mathbf{x}) = (z_1(\mathbf{x}), \ldots, z_S(\mathbf{x}))$.

Although the costs associated with the stage one decision are fixed, they are included in the scenario objective functions to produce the scenario objective function values as at the start of stage one. This allows the criterion vectors of different stage one decisions to be compared.

For any choice of the stage one decision vector, $\mathbf{x}$, there is a single optimal objective function value for each scenario, and so, for each integer realisation of the stage one decision vector there is a single point in criterion space. This means that both the set of feasible stage one decisions, and the noninferior set are discrete. It is, conceptually, if not computationally, possible to find the noninferior set by carrying out a full enumeration of the binary values of the stage one decision vector $\mathbf{x}$. Full enumeration can be avoided by using a branch and bound procedure and a suitable algorithm is developed in this chapter. First, however, the issue of unsupported noninferior criterion vectors will be discussed.

7.3 Unsupported, Nondominated Criterion Vectors

This issue is discussed in Steuer (1986), and we summarise the main points here to set the scene. Figure 7.1 has been adapted from this book.

Let X be the set of stage one decisions that are feasible in problem (7.1).

Let $X^{\leq}$ be the feasible region of the linear relaxation of (7.1) in which the integrality requirements are replaced by the constraint $0 \leq \mathbf{x} \leq 1$.

We will denote the feasible set of (7.1) in criterion space as Q, and the feasible

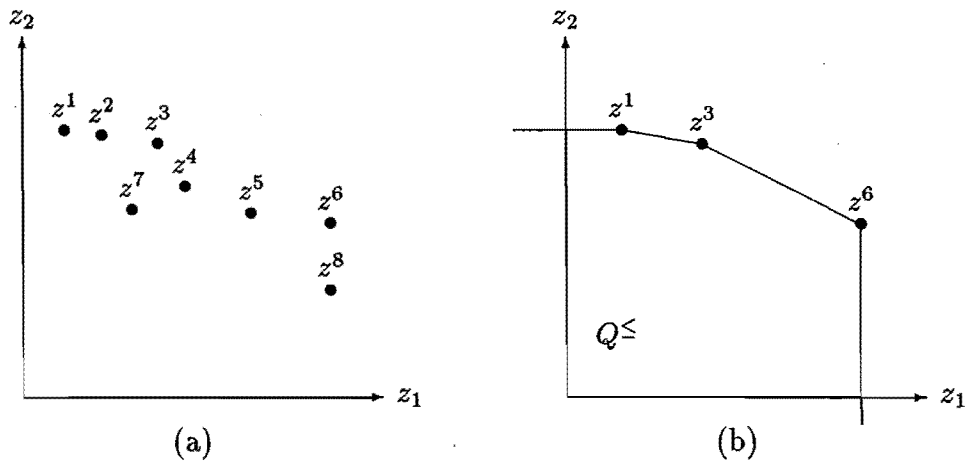

Figure 7.1: Supported and Unsupported Points in Criterion Space

region of the linear relaxation as $Q^{\leq}$. The noninferior set is denoted as N. That is:

$$Q = \{\mathbf{z}(\mathbf{x})|\mathbf{x} \in X\}$$
$$Q^{\leq} = \{\mathbf{z}(\mathbf{x})|\mathbf{x} \in X^{\leq}\}$$

and

$$N \in Q \in Q^{\leq}$$

We have the following definitions:

Definition 7.1 *Let* $\mathbf{z} \in \mathbf{N}$. *Then if* $\mathbf{z}$ *is on the boundary of* $Q^{\leq}$, $\mathbf{z}$ *is a convex* supported *nondominated criterion vector. Otherwise,* $\mathbf{z}$ *is an* unsupported *(*convex dominated*) nondominated criterion vector.*

Definition 7.2 *Let* $\mathbf{z} \in \mathbf{N}$ *be supported. Then* $\mathbf{z}$ *is* supported-extreme *if it is an extreme point of* $Q^{\leq}$. *Otherwise,* $\mathbf{z}$ *is* supported-nonextreme.

For example, let $Z = \{\mathbf{z}^i|1 \leq i \leq 8\}$ in Figure 7.1(a). Note that $\mathbf{z}^2$ is a convex combination of $\mathbf{z}^1$ and $\mathbf{z}^3$. As shown via Figure 7.1(b), of the nondominated criterion vectors:

a) $\mathbf{z}^1$, $\mathbf{z}^3$, $\mathbf{z}^6$ are supported-extreme.

b) $\mathbf{z}^2$ is supported-nonextreme.

c) $\mathbf{z}^4$, $\mathbf{z}^5$ are unsupported.

The terms are not applied to $\mathbf{z}^7$ or $\mathbf{z}^8$ because both vectors are dominated.

7.3.1 Weighted-Sums Approaches and Unsupportedness

Unsupported, nondominated criterion vectors do not occur in MOLPs. However, in multiple objective integer programs, unsupported, nondominated criterion vectors are a likely occurrence. Because each unsupported member of N is dominated by some convex combination of other members of N, it is not possible to generate unsupported criterion vectors using the weighted-sums program:

$$\max\left\{\sum_{i=1}^{k} w_\omega z_\omega(\mathbf{x}) \mid \mathbf{x} \in X\right\}$$

regardless of the choice of $\mathbf{w}$.

For example, let the problem of Figure 7.1(a) be a problem with integer stage one variables such that the points $\{\mathbf{z}^i | 1 \leq i \leq 6\}$ are noninferior. A weighted-sums method (such as NISE) can only generate the supported-extreme points $\mathbf{z}^1$, $\mathbf{z}^3$, and $\mathbf{z}^6$.

This has implications for integer stochastic optimisation because the objective function of a stochastic programme is the weighted sum of the scenario objectives. This means that a stochastic programme cannot find unsupported extreme points. While this does not matter if the decision maker's only concern is to find a solution that maximises the expected value over the scenarios, it does mean that a stochastic integer programme cannot report unsupported nondominated solutions, and so there may be alternative courses of action that the decision maker cannot discover using stochastic optimisation.

For example, consider Figure 7.1(a) and assume that four points have the following coordinates: $z^1 = (8, 38)$, $z^3 = (20, 35)$, $z^4 = (24, 30)$, $z^5 = (33, 27)$, $z^6 = (40, 25)$, and that scenario 1 has probability p_1, and scenario 2 has probability p_2. Then, for $0 \leq p_1 \leq 0.2$, z^1 produces the maximum expected value. From $0.2 \leq p_1 \leq 0.33$, z^3 produces the maximum expected value, while for $0.33 \leq p_1 \leq 1$, z^6 produces the maximum expected value. This means that as soon as the probability of scenario 1 goes above 0.33, the solution presented to the decision maker switches from z^3 which performs very well under scenario 2, to z^6 which is optimal for scenario 1.

Neither of the intermediate solutions, z^4 or z^5, that provide more balance between the scenarios are made known to the decision maker. This is irrelevant if the decision maker's sole concern is to maximise the expected value, because the expected values of z^4 and z^5 are lower than the expected value of z^6 for $0.33 \leq p_1 \leq 1$. However, if the decision maker has additional concerns, such as avoiding large differences between the scenario outcomes, then it is of considerable value to know that z^4 and z^5 exist.

7.4 Multicriteria Branch and Bound

The branch and bound procedure proposed in this section is based on the determination of an upper bounding criterion vector at the current node of the branch and bound tree. If the upper bounding criterion vector is dominated by the criterion vector of a known integer solution, then every integer solution below the node must be dominated, and the node can be fathomed.

At a particular node, some of the stage one variables are fixed at binary values, while the other stage one variables (referred to as free variables) are allowed to be continuous, but restricted to be in the range 0 to 1. The nonanticipativity restrictions on the free stage one variables are relaxed and the subproblem is solved for each scenario independently. The resulting scenario objective function values are upper bounds on the scenario objective function values of any integer solution found further down the tree, and the vector of these scenario objective function values forms an upper bound for the criterion vector of any integer solution found further down the tree.

If this upper bound is dominated by the criterion vector of an integer solution that has already been found, then the node is fathomed. Otherwise, one of the free variables is fixed to be zero or one, and a new node is created at the next level down the tree.

7.4.1 Fathoming a Node

Consider a node in which some of the stage one variables have been fixed at either zero or one, and the remaining variables are free variables. We will denote the fixed (binary valued) variables as $\mathbf{x}^B$ and the free variables as $\mathbf{x}^F$, so that $\mathbf{x} = \mathbf{x}^B \cup \mathbf{x}^F$.

At such a node, problem (7.1) becomes a set of S subproblems, one for each scenario. The subproblem for scenario ω will be problem (7.3):

$$\begin{aligned}
\text{maximise} \quad & z_\omega(\mathbf{x}^B) = \mathbf{q}_\omega \mathbf{y}_\omega + (\mathbf{c}_0^F + \mathbf{c}_\omega^F)\mathbf{x}_\omega^F + (\mathbf{c}_0^B + \mathbf{c}_\omega^B)\mathbf{x}^B && (7.3\,\text{a}) \\
\text{subject to} \quad & A^F \mathbf{x}_\omega^F = \mathbf{b}_0 - A^B \mathbf{x}^B && (7.3\,\text{b}) \\
& B_k \mathbf{y}_k + T_k^F \mathbf{x}_k^F = \mathbf{b}_k - T_k^B \mathbf{x}^B \quad \forall\ k \in \Omega && (7.3\,\text{c}) \\
& \mathbf{y}_\omega \geq 0 \\
& 0 \leq \mathbf{x}_\omega^F \leq 1
\end{aligned}$$

where: c_0^F = the stage one costs and benefits corresponding to the free variables

c_0^B = the stage one costs and benefits corresponding to the fixed variables

c_ω^F = the stage two costs corresponding to the free variables

c_ω^B = the stage two costs corresponding to the fixed variables

A^F = the columns of A corresponding to the free variables

A^B = the columns of A corresponding to the fixed variables

T^F = the columns of T corresponding to the free variables

T^B = the columns of T corresponding to the fixed variables

$\mathbf{x}_\omega^B$ = the variables that have been fixed at binary values

$\mathbf{x}_\omega^F$ = the free variables, which are optimised for scenario ω

$\mathbf{y}_\omega$ = the optimal recourse variables under scenario ω

$\mathbf{y}_k$ = the optimal recourse variables under scenario k, given $\mathbf{x}_\omega^F$

Constraints (7.3 b) require that the free variables be feasible in stage one, and constraints (7.3 c) require that the free variables be feasible in stage two under all scenarios. The fixed variables are moved to the right hand side in both sets of constraints.

Proposition 7.1 *When subproblem* (7.3) *is solved at a node for a particular scenario, the optimal objective function value is an upper bound on the objective function value for that scenario for all solutions below the node in the tree.*

Proof: If we consider a single scenario in isolation, then problem (7.3) is a relaxation of a mixed integer problem with a single objective, and the proof follows

directly from the standard branch and bound theory.

Proposition 7.2 *When problem* (7.3) *is solved at a node for every scenario, the resulting criterion vector,* $\mathbf{z}(\mathbf{x}^B) = (z_1^*(\mathbf{x}^B), z_2^*(\mathbf{x}^B), \ldots, z_S^*(\mathbf{x}^B))$, *is an upper bound on the criterion vectors for all solutions below the node in the tree.*

Proof: Let $\hat{\mathbf{x}}^{\mathbf{B}}$ be the fixed variables at a node in the tree below the current node, and let the criterion vector of these variables be $\hat{\mathbf{z}}(\hat{\mathbf{x}}^{\mathbf{B}}) = (\hat{z}_1^*(\hat{\mathbf{x}}^{\mathbf{B}}), \hat{z}_2^*(\hat{\mathbf{x}}^{\mathbf{B}}), \ldots, \hat{z}_S^*(\hat{\mathbf{x}}^{\mathbf{B}}))$.

If we consider a single scenario, ω, then, by Proposition 7.1, $z_\omega^*(\mathbf{x}^B) \geq \hat{z}_\omega^*(\hat{\mathbf{x}}^{\mathbf{B}})$. From this it follows that criterion vector $\mathbf{z}(\mathbf{x}^B) \geq \hat{\mathbf{z}}(\hat{\mathbf{x}}^{\mathbf{B}})$, and the proposition is proved.

Proposition 7.3 *A criterion vector that dominates the upper bounding criterion vector of a node in the branch and bound tree also dominates the criterion vectors of all integer solutions that occur at nodes further down the branch and bound tree.*

Proof: Let $\bar{\mathbf{z}}$ be a criterion vector that dominates the upper bounding criterion vector, $\mathbf{z}(\mathbf{x}^B)$, of some node in the branch and bound tree. That is, $\bar{z}_\omega^* \geq z_\omega^*(\mathbf{x}^B)\ \forall\, \omega \in \Omega$ with $\bar{z}_\omega^* > z_\omega^*(\mathbf{x}^B)$ for at least one of $\omega \in \Omega$.

Let $\hat{\mathbf{z}}(\hat{\mathbf{x}}^{\mathbf{B}})$ be the criterion vector of a node lower down the tree at which the solution is integer. Since, by Proposition 7.2, $\mathbf{z}(\mathbf{x}^B) \geq \hat{\mathbf{z}}(\hat{\mathbf{x}}^{\mathbf{B}})$, it follows that $\bar{z}_\omega^* \geq \hat{z}_\omega^*(\hat{\mathbf{x}}^{\mathbf{B}})\ \ \forall\ \ \omega \in \Omega$ with $\bar{z}_\omega^* > \hat{z}_\omega^*(\hat{\mathbf{x}}^{\mathbf{B}})$ for at least one of $\omega \in \Omega$, and the proposition is proved.

7.4.2 The Fathoming Rules

The rules used to fathom a node are similar to the fathoming rules of the single objective branch and bound. That is:

1. Infeasibility. If problem (7.3), the linear relaxation at a node, is infeasible for one, or more, of the scenarios, then the node can be fathomed.

2. Integrality. The integrality requirements are satisfied when the end of a branch has been reached and all stage one variables have been fixed at binary values. Integrality is also satisfied at a node if the subproblem for every scenario produces the same optimal stage one decision vector, and all stage one variables are binary.
   That is, in problem (7.3), if $\mathbf{x}_j^F \in \{0,1\}$ and $\mathbf{x}_i^F = \mathbf{x}_j^F\ \ \forall\, i,j\ \in\ \Omega$.

3. Domination. When the upper bounding criterion vector at a node, $\mathbf{z}(\mathbf{x}^B)$, is dominated by the criterion vector of a known integer solution, then, by Proposition 7.3, it is not possible for a nondominated integer solution to be found further down the tree, and the node can be fathomed.

4. The objective function value of a scenario is less than a specified minimum. If the decision maker has specified minimum acceptable values for the objective functions of some, or all, of the scenarios, then a node can be fathomed if the objective function value of a scenario is less than that value. If, for problem (7.3), $z^*_\omega(\mathbf{x}^B) < z^m_\omega$, where z^m_ω is the minimum acceptable objective function value for scenario ω, then, by Proposition 7.1, the objective function value for scenario ω cannot attain the value z^m_ω at any node further down the tree, and the node can be fathomed.
   Note: this means of fathoming a node could be implemented by appending the constraints $z_\omega(\mathbf{x}^B) \geq z^m_\omega \quad \forall\, \omega \in \Omega$ to problem (7.3), in which case failure to reach a minimum objective function value would make the subproblem infeasible.

7.4.3 Branching from a Node

The variable to be branched on is chosen with the aim of maintaining the objective function values of as many of the scenarios as possible, to increase the likelihood that the integer solution that is reached will be nondominated. Nondominated solutions are desirable because identification of nondominated solutions early in the branch and bound algorithm increases the likelihood that upper bounding criterion vectors found subsequently can be shown to be dominated, thus fathoming the corresponding nodes.

The branching variable is chosen by searching the $\mathbf{x}^F_\omega$ vectors for a decision variable that is "preferred" by a majority of the scenarios. The attractiveness of a free variable for use as the branching variable is greatest when its value is close to being binary, that is, close to zero or one. Intuitively it would seem that, for any scenario, the smaller the change in the value of the variable when it is fixed at a binary value, the smaller the resulting reduction in the objective function value for that scenario. Also, a variable that can be fixed at a binary value with small reductions in objective function values for the greatest number of scenarios is more

likely to lead to a nondominated integer solution than a variable that leads to large reductions in objective function values when it is fixed. However, to be attractive as the variable to fix, a variable must be close to the same bound (0 or 1) for several scenarios. Clearly, a variable that has a value close to 1 for two scenarios, and close to 0 for two scenarios, would not be as good a choice as one that is close to 1 (or 0) for four scenarios.

An operational advantage of using this approach is that it can reduce the number of subproblems that must be solved. If the branching variable at a node has a binary value in the solution for a scenario, and the variable is fixed at that binary value when branching to the next node, then the subproblem at the next node does not have to be solved for that scenario.

The algorithm identifies the free variable that has a value of 1 in the optimal solutions of the greatest number of scenarios, and the free variable that has a value of 0 in the optimal solutions of the greatest number of scenarios. The variable with the larger count is chosen for branching. If the count is the same for 1 and for 0, the tie is arbitrarily broken by choosing the variable that has a value of 1 most often.

If no free variable has a binary value in any of the scenario solutions, then the variable closest to having a binary value in a scenario solution is chosen. Again, ties are broken by selecting the variable closest to 1, rather than the variable closest to 0.

7.5 The Solution Method

The solution method makes a sequence of passes through the branch and bound tree. In the first pass, the branch and bound tree is defined and the first noninferior criterion vector found. The scenario-maximal solutions are found in the second pass. In the third, and final pass, a complete fathoming of the branch and bound tree is carried out to find the rest of the noninferior set.

These three passes are described in the following sections.

7.5.1 Pass One: Determine the Branch and Bound Tree

The first step is to solve the full linear relaxation of the problem for all scenarios. If the full relaxation is infeasible for any scenario, then the whole problem is infeasible

and the procedure stops.

The first pass has two objectives. The first is to determine a branch and bound tree for the problem, and the second is to find an integer solution that provides, in a balanced fashion, for all of the scenarios. That is, the integer solution found by the first pass should correspond to a criterion vector that is central in objective space, rather than one that is close to an edge. That is, the criterion vector does not include extreme trade-offs between the scenarios. This will usually mean that no scenario objective function value is close to the scenario-maximal objective function value for that scenario. The idea is that this criterion vector, plus the scenario-maximal criterion vectors found in the second pass, will provide a first approximation to the noninferior set, and may be sufficient to prove the inferiority of many criterion vectors generated in the final pass. What we are looking for is a set of criterion vectors that can prove inferiority high in the branch and bound tree, and so make it possible to fathom nodes high in the tree.

The branch and bound tree is described as a series of levels, one for each of the binary variables. A branching variable is specified for each level, and this variable will be fixed at either 0 or 1 when proceeding to the next level. At a node at any particular level the branching variables of the levels above are fixed variables, while the variable that is to be branched on at this level, and the variables that will be branched on at the levels below it, are free variables.

Many problems with binary variables include restrictions on the combinations of values that can be included in a solution. For example, it may not be possible to build both of two types of plant, or it may be that at least one of several types of plant must be built. The existence of logical restrictions of this type provide the opportunity to prune the branch and bound tree, and this pruning is most effective if these restrictions are enforced high in the tree. For this reason the algorithm can be given a list of variables that have logical restrictions, and these variables are used first when building the branch and bound tree. For example, if no more than one of two variables is allowed take a value of 1 in any solution, then branching on those two variables at the first two levels in the tree will allow 1/4 of the tree to be removed. That is, combinations (0,0), (1,0) and (0,1) are feasible, but (1,1) is not. This eliminates one of the four branches at the second level in the tree.

At levels in the tree corresponding to variables that are included in the logical restrictions the variable to branch on is decided, and the only decision is whether

to fix its value at 0 or 1. For the remaining levels, the variable must be chosen and the branching value determined. This process is discussed in Section 7.6, in which the full algorithm is described.

The first pass is complete when the bottom of the branch and bound tree has been reached, and a feasible integer solution has been found. At this point the branch and bound tree will have been defined.

7.5.2 Pass Two: Find the Scenario-Maximal Criterion Vectors

In the second pass the problem is solved as a single objective problem for each scenario in turn. The object here is to find the scenario-maximal solutions and their corresponding criterion vectors. These criterion vectors are reported back to the decision maker, to help him/her set lower bounds on the scenario objective function values. The resulting lower bounds are used in the fathoming rules of the final pass.

The procedure takes each scenario in turn. For each scenario the best objective function value is selected from the list of integer criterion vectors that have been found so far. This value is used as the initial incumbent objective function value for the scenario. For the first scenario processed this list contains the single criterion vector found in stage one. For subsequent scenarios the list will also contain the scenario-maximal criterion vectors of the scenarios already considered, plus the criterion vectors of any other integer solutions that were found while searching for the scenario-maximal vectors.

The procedure follows the branch and bound tree that was determined during the first pass. Although this will generally not be the most efficient tree for the particular scenario, a complete branch and bound tree must eventually be analysed in order to find the whole noninferior set. By using the same tree in every pass, the solutions to the subproblems can be stored for use at subsequent passes. Reuse of solutions should reduce the number of subproblems that have to be solved from the number that would be required if a different tree were followed for each scenario.

At each node in the branch and bound tree the subproblem is solved for the scenario under consideration only, and the usual single objective fathoming rules are used.

At the bottom of the tree an integer node has been found. If the objective

function value for the scenario is greater than, or equal to, the incumbent value, then the subproblems are solved for all of the scenarios. If the objective function value is less than the incumbent value, then the subproblems are not solved for the other scenarios. Although this is an integer solution that may turn out to be noninferior, its criterion vector is not determined at this stage. The primary aim of the second pass is to find the scenario-maximal solutions, and so computations that can be deferred until later are deferred. If necessary, this criterion vector will be found in the third pass, but if it is, in fact, dominated, then some work will have been avoided by not solving for the other scenarios at this point.

If the criterion vector of this integer solution dominates the incumbent criterion vector, then this solution becomes the incumbent, otherwise, the old incumbent is retained. This check is required when the objective function value for this scenario is equal to that of the incumbent. Without the check, it is possible for a dominated solution to be selected as the scenario-maximal solution.

An input parameter instructs the algorithm whether to continue the search for the true scenario-maximal solution, or whether to accept the one found when the bottom of the tree is first reached. If the true scenario-maximal solution is to be found, then the algorithm continues by backing up the tree. Otherwise, it exits for the next scenario.

7.5.3 Determine Lower Bounds For The Scenario Objectives

When the scenario-maximal solutions have been found for all scenarios, the corresponding criterion vectors can be reported back to the decision maker. The decision maker can examine these vectors to determine the minimum acceptable objective function value for each scenario. These minimum acceptable values are then used as lower bounds on the scenario objective function values during the search for the rest of the noninferior set. At any node, if the objective function value of a scenario solution to the subproblem is less than the minimum acceptable value, then, by proposition 7.1, no solution further down the tree can produce an objective function value for the scenario that is greater than the minimum acceptable value. This means that the node can be fathomed.

Use of lower bounds on the scenario objective function values means that parts of the noninferior set will not be found. However, by providing the lower bounds,

the decision maker has declared that these solutions are of no interest, and so there is no point in describing these parts of the noninferior set.

Alternatively, the algorithm can be instructed at the start to use a specified fraction of the minimum objective function values as the lower bounds, in which case no interaction with the decision maker is required.

7.5.4 Pass Three: Find The Whole Noninferior Set

The branch and bound tree is now followed to find the whole of the noninferior set.

The first step at each node is to check whether the node has been visited in any of the previous stages. If it has, then there will be a known solution to the subproblem for at least one scenario. The node can be fathomed if, for a known scenario solution, the objective function value for the scenario is less than its lower bound. If the node cannot be fathomed by considering the known scenario solutions, then the subproblem is solved for each of the remaining scenarios. After the subproblem has been solved for a scenario the objective function value is compared to the lower bound for that scenario, and the node fathomed if possible. Once the subproblem has been solved for all scenarios, the full upper bounding criterion vector for the node has been found. If this vector is dominated by the criterion vector of a known integer solution, then the node is fathomed. If the upper bounding criterion vector is not dominated, then the algorithm decides whether to fix the branching variable at 1 or 0, as described in Section 7.4.3. If, in the solution to the subproblem for any scenario the branching variable is at the value being branched on, then the solution and objective function value for that scenario are carried down to the next node.

The algorithm continues down the branch and bound tree until an integer node is reached, or the current node can be fathomed. It then backs up the tree to the previous node and follows the other branch if the node has not yet been fathomed. If it has been fathomed the algorithm backs up to the node above that. Eventually, the algorithm backs up to the top level of the tree for the second time and the process terminates.

7.5.5 The Final Step: Reduce To The True Noninferior Set

Stage four of the procedure removes from the list of integer solutions those that are, in fact, dominated. As integer solutions were found in the previous stages they were

checked to see if they were dominated by any of the solutions in the current list of known integer solutions, and only added to the list if they were nondominated. However, some of the solutions in the list may, in fact, be dominated by solutions that were found later. To finally determine the noninferior set, all of the solutions in the list must be checked against all of the other solutions, and any dominated ones discarded.

7.6 The Solution Algorithm

In this section we will describe the solution algorithm as a series of routines.

7.6.1 Branching When the Branching Variable is Not Yet Known - First Pass Only

During the first pass the variables to branch on have yet to be determined, and the decisions about which variable to branch on, and whether it should be fixed at 0 or 1, are taken together.

First, the variables that are fixed at this level are checked against the logic rules. If any rule is violated, then the node is infeasible and the procedure backs up the tree to the previous level. If the logic rules are satisfied, the subproblem is solved for each scenario. A count is taken of the number of times in the scenario solutions to the subproblem that each free variable has a value of 1, and that it has a value of 0. The variable with the largest count is chosen as the branching variable. If the greatest count was for being at 1, then the branching variable is fixed at 1, if it was for being at 0, then the branching variable is fixed at 0. In the case of a tie, 1 is preferred to 0. If no variable has a count greater than zero, then, for each free variable, in each scenario solution to the subproblem, the difference between its value and the nearer binary bound is found. The branching variable is chosen to be the one with the smallest difference, and it is fixed at the closer bound. If, in the solution to the subproblem for any scenario the branching variable is at the value being branched on, then the solution and objective function value for that scenario are carried down to the next node.

Instead of comparing two values to determine whether they are exactly equal, they can be tested to see if they are approximately equal by specifying a tolerance.

The tolerance to use for testing whether the values of free variables should be considered to be binary is given as an input parameter at the start. When the tolerance is set to large values, the algorithm will treat more variables as being binary than when it is set to small values. This will lead the algorithm to copy more solutions down to the next node, and thus reduce the number of subproblems that must be solved at the next node. However, this will also lead to the upper bounding criterion vector at the next node being overstated, which may mean that the node is not fathomed when it could have been. This approximation only affects the objective function values for the scenarios that had solution values that were not exactly binary, and it will be corrected as soon as subproblems are solved for those scenarios at lower levels in the tree.

If variables are fixed to be binary using a large tolerance, and the scenario solution to a subproblem is carried all the way down to the bottom of the tree, then the scenario objective function value in the integer criterion vector may be overstated. To ensure that the criterion vector of the integer solution is correct the subproblem at the bottom of the tree is solved for all scenarios. This will be relatively cheap, computationally, because all of the binary variables are fixed, and only the stage two LP's have to be solved.

The pseudo code for this process follows:

Step 1 Check the feasibility of the node

If the fixed variables violate a logic rule, then

- fathom the node and backup: call routine in Section 7.6.4
- go to Step 1

End If

Step 2 Add this node to the list of nodes already seen

Step 3 Obtain scenario solutions at this node

For each scenario

If a solution for the scenario was carried down, then

If this is the bottom level, then

- use the solution as the initial solution for the solve
- **otherwise**
- don't solve for this scenario: loop for the next scenario

End If

- solve subproblem (7.3) for the scenario

If the subproblem is infeasible, then

- flag the node as infeasible
- fathom the node and backup: call routine in Section 7.6.4
- go to Step 1

End If

End If

– store the objective value and the stage one decision vector

– the recourse decision vector is not required again, so don't store it

End For

Step 4 Exit the routine as soon as an integer solution is found

If this node is at the bottom level, then

- we have a feasible integer solution, and are finished
- store the integer solution and its criterion vector
- Exit the first pass

End If

Step 5 Select a free variable to branch on:

– let the tolerance for testing if a value is binary be α

For each free variable, $x \in \mathrm{x}^F$

– count the number of scenario solutions in which $x \geq (1-\alpha) \rightarrow c_1(x)$

– count the number of scenario solutions in which $x \leq \alpha \rightarrow c_0(x)$

– set $d_1(x)$ to the minimum over all scenario solutions of $(1-x)$

– set $d_0(x)$ to the minimum over all scenario solutions of x

End For

Find the largest count of binary values: $c_{max} = \max_{x \in \mathrm{x}^F}(\max(c_1(x), c_0(x)))$

If $c_{max} > 0$, then

- use the corresponding variable as the branching variable, call it x^b

If $c_1(x^b) \geq c_0(x^b)$, then

- fix $x^b = 1$
- otherwise, fix $x^b = 0$

End If

End If

If $c_{max} = 0$, that is, no free variable has a binary value in any scenario solution, then

- find the minimum distance from being binary:

  $d_{min} = \min_{x \in \mathbf{x}^F}(\min(d_1(x), d_0(x)))$

- use the corresponding variable as the branching variable, call it x^b

  If the value of x^b was closest to 1, then

  - fix $x^b = 1$
  - otherwise fix $x^b = 0$

  End If

End If

Step 6 Initialise the list of free variables, $\mathbf{x}^F$, for the next level in the tree

If the variable to branch on is determined by the logic rules, then

- initialise the list of free variables as that variable
- **otherwise,**
- initialise the list of free variables as those not fixed higher in the tree

End If

Step 7 Remove from the list of free variables those already fully branched

- initialise the possible branches as $\{0, 1\}$ for every free variable

  For all free variables, $x^f \in \mathbf{x}^F$

  - build the node that would be branched to if x_f was fixed at 1

    If the node is in the list of nodes already seen, then

    - remove 1 from the set of possible branches for x_f

    End If

  - build the node that would be branched to if x_f was fixed at 0

    If the node is in the list of nodes already seen, then

    - remove 0 from the set of possible branches for x_f

    End If

    If the set of possible branches for x_f is empty, then

    - remove x_f from the list of free variables for the node

    End If

  End For

Step 8 Continue to the next node, or backup

If $\mathbf{x}^F$ is empty, then

- fathom the node and backup: call routine in Section 7.6.4
- go to Step 1

End If

- carry scenario solutions down: call routine in Section 7.6.3

Step 9 go to Step 1

7.6.2 Branching When the Branching Variable is Known

In passes two and three, the variable to branch on at each node is already known, and the only decision is whether to fix it at 0 or 1. If the node has already been branched from, then the criterion vector and subproblem solutions are already known, and there is no need to solve the subproblems again. The variable is fixed to the binary value not used last time and the next node created.

If it is the first time we are branching from this node, then the choice of branch is made in the following manner. First, the subproblem is solved for each scenario. Then a count is taken of the number of times the branching variable has a value of 1, and the number of times it has a value of 0 in the solutions to the subproblems. If either count is greater than 0, then the branching variable is fixed to the binary value with the larger count. To break a tie, 1 is used. If both counts are zero, then, for each of the subproblem solutions, the difference between the value of the branching variable and the nearer binary bound is found. The branching variable is fixed at the bound that gave the smallest difference. As in the first pass (see Section 7.6.1) a tolerance is used to when determining whether the value of a variable should be considered to be binary.

The pseudo code for this process follows:

Step 1 Check the feasibility of the node

If the fixed variables violate a logic rule, then

- fathom the node and backup: call routine in Section 7.6.4
- go to Step 1

End If

If the node already exists and is infeasible, then

- fathom the node and backup: call routine in Section 7.6.4
- go to Step 1

End If

Step 2 Special case at the bottom level

If this node at the bottom level, then

– have an integer solution, and no branching is left to be done

– set list of scenarios to be solved to be all scenarios

– go to Step 6

End If

Step 3 Read the branching variable from the level data

– let the branching variable be x^b

Step 4 Determine whether the subproblem should be solved

If this node has already been branched, then

If both branches have already been taken (0 and 1), then

- fathom the node and backup: call routine in Section 7.6.4
- go to Step 1
- **otherwise,**
- fix the branching variable to take the other branch
- go to Step 9 (the subproblems have already been solved)

End If

End If

If carrying out the second pass, then

– set the list of scenarios to be solved to be the current scenario

– **otherwise,**

– set the list of scenarios to be solved to be all scenarios

End If

Step 5 Add this node to the list of nodes already seen

Step 6 Obtain scenario solutions to the subproblem

For each scenario in the list of scenarios to be solved

If a solution for the scenario was carried down, then

If this is the bottom level, then

- use the solution as the initial solution when solving
- **otherwise**
- don't solve for this scenario - loop for the next scenario

End If

- solve subproblem (7.3) for the scenario

If the subproblem is infeasible, then

- flag the node as infeasible
- fathom the node and backup: call routine in Section 7.6.4
- go to Step 1

End If

End If

– store the objective value and the stage one decision vector

– the recourse decision vector is not required again, so don't store it

End For

Step 7 Processing at the bottom level

If if this node is at the bottom level, then

– we have a feasible integer solution

– store the integer solution and its criterion vector

If finding scenario-maximal solutions (the second pass), then

If stopping at the first integer solution for each scenario, then

- Exit for the next scenario

End If

End If

– fathom the node and backup: call routine in Section 7.6.4

End If

Step 8 Determine the value to branch on (Not at the bottom level.)

– let the tolerance for testing if a value is binary be α

– count the number of scenario solutions in which $x^b \geq (1-\alpha) \rightarrow c_1$

– count the number of scenario solutions in which $x^b \leq \alpha \rightarrow c_0$

– set $d_1(x)$ to the minimum over all scenario solutions of $(1-x^b)$

– set $d_0(x)$ to the minimum over all scenario solutions of x^b

If $c_1 \geq c_0$ and $c_1 > 0$, then

– fix $x^b = 1$

End If

If $c_1 < c_0$ and $c_0 > 0$, then

- fix $x^b = 0$

End If

If $c_1 = c_0 = 0$ and $d_1 < d_0$, then

- fix $x^b = 1$
- otherwise, fix $x^b = 0$

End If

Step 9 Move to the node at the next level in the tree

- carry scenario solutions down: call routine in Section 7.6.3

Step 10 go to Step 1

7.6.3 Copy Subproblem Solutions Down One Level in the Tree

Some housekeeping is required each time the algorithm moves to another node in the tree, both when branching down, and when backing up. As previously discussed, a node may have scenario solutions in common with the node above it. When this is the case, these scenario solutions should be copied down to the current node from the node above.

Step 1 Copy down solutions from the node above

- let the variable branched on at the node above be x^b
- let the value it was fixed to be I
- let the value of x^b in the solution for scenario ω at the node above be x^b_ω

For each scenario, $\omega \in \Omega$

If $x^b_\omega = I$, then

- copy to the current node from the node above:
  - the solution vector for this scenario, $\mathbf{x}_\omega$
  - the objective value for this scenario, z_ω

End If

End For

Step 2 **Return**

7.6.4 Backing up to a Node

When a node has been fathomed, the algorithm backs up to the previous node in the tree. The required housekeeping is somewhat different when backing up to a node, rather than branching down to it. The algorithm again determines which variables are available for branching. If there are no branches from the node that have not already been taken, then the node is fathomed, and the algorithm backs up another level in the tree. When the top level in the tree is backed up to for the second time, the tree has been fully fathomed and the process stops.

Step 1 Unfix the branching variable

Step 2 Update the list of free variables

If the branching variable has been fixed at 0 and at 1, then
- both branches have been fathomed for this variable
- so remove it from the list of free variables for this node

End If

Step 3 Determine whether to branch or backup again

If the list of free variables is empty, then
- all possible branches have been fathomed
  If this is the top level node, then
  - the tree is fully fathomed
  - **STOP**

  End If
- backup to the previous node, go to Step 1
- **otherwise,**
- branch again from this node
- carry the scenario solutions down: call routine in Section 7.6.3
- **Return**

End If

7.7 Summary

In this chapter we have extended our work to consider planning problems in which the stage-one decision variables are binary, and the recourse decision variables are continuous. That is, the problem consists of a set of choices at stage one, each of which can be taken, or not taken, and a set of operating decisions at stage two. This problem structure approximates the structure of many strategic planning problems in which investment decisions taken now will determine the organisation's operating parameters in the future. This problem is a mixed integer problem, and the presence of several scenarios makes it a multiobjective optimisation problem.

We have developed a branch and bound algorithm suitable for finding the efficient solutions to the problem for the decision maker to choose among. Finding the noninferior set for a mixed integer, multiobjective problem is much more computationally demanding than finding the noninferior set for a multiobjective problem with continuous variables. It is also more computationally demanding than finding the optimal solution for an MIP with a single objective. Rather than being a scalar, the incumbent solution is a list of criterion vectors that have not been proved to be dominated. At each node in the branch and bound tree a criterion vector must be calculated, rather than a single objective value, and this vector must be compared to the list of incumbents in order to fathom the node. These complexities increase the computational effort, and the storage requirements, compared to problems with a single objective. We have been able to reduce the computational effort to some degree by taking advantage of the problem structure.

The Tchebycheff approach discussed in Section 3.4 can be used to find noninferior points for problems with integer variables. If the intention was solely to find a decision to implement, then this approach would be applicable here. However, the intention of this work is to provide decision makers with a set of solutions to choose among, and to produce insights about the problem and the available alternatives. The Tchebycheff approach must use a mixed integer solution method to find the criterion vectors to present to the decision maker. This is likely to be an implementation of the branch and bound algorithm, in which case it would be more efficient to apply the branch and bound algorithm developed here. This algorithm can be used to find the approximation to the noninferior set all at once, whereas the Tchebycheff approach would have to analyse the branch and bound tree to optimality for every

criterion vector presented to the decision maker.

The development of this algorithm extends the applicability of our approach to decision making under uncertainty to a problem structure that is well suited to modelling strategic planning problems. That is, to situations in which the decision maker must choose between alternative courses of action that will decide the shape of the organisation for some years to come. These choices are binary in nature because each alternative must either be selected or not selected. However, once an alternative has been selected, the resulting operational problem is continuous.

As mentioned in Section 3.5, Mavrotas, Diakoulaki & Papayannakis (1999) report a branch and bound algorithm for solving mixed integer multiobjective optimisation problems that is similar to the algorithm developed in this thesis. However, this paper came to our notice after we had completed our work. We also note that the example reported by Mavrotas et al. (1999) has only two objectives, whereas we have applied our algorithm to a problem with five objectives.

In Chapter 8 we report the application of this algorithm to an example problem, and show how the nonferior set can be described to decision makers.

Chapter 8

A Capacity Expansion Problem

8.1 Introduction

In this chapter we apply the algorithm developed in Chapter 7 to an example problem in which a company wants to enter an electricity retail market as an electricity supplier. It can build hydro generating capacity, thermal capacity, or a combination. The lead times are such that the company must start its building programme now, in order to be ready to enter the market in ten years' time. The size of this future market will be determined by the economic conditions over the next decade, and whether the government introduces a carbon tax. A set of five scenarios is developed to describe what the market may look like in ten years' time. The algorithm produces a set of 63 efficient stage-one decisions for the company to choose among. We briefly describe this set, and discuss how decision makers could interpret it to gain insights about the problem and to find a decision to implement.

8.2 The Problem

We consider a company, ABC Inc, that is planning to enter an electricity supply market. The company has the necessary water rights to develop a section of river for hydro generation. It is also in a position to build thermal generating stations at various sites to burn gas, coal or lignite. The company believes that it can compete for a portion of the current electricity demand, and for future increases in demand. The demand for electricity follows the seasonal weather pattern of the region. There is a wet season which generally lasts about five months, and a dry season that lasts

about seven months. Electricity demand is higher in the dry season than in the wet season. However, river flows are much greater in the wet season than in the dry. The company's water rights permit the company to store water during the wet season and to use it for electricity generation in the dry season.

The demand for electricity follows the usual pattern of a base load, taken all of the time, a middle load demanded for 60% of the time, and a peak load taken for 20% of the time. There is also a large electricity user, XYZ, that requires continuous supply, and buys its power by contract outside the general electricity market. XYZ is talking of expanding its operations, in which case it would require a large quantity of additional power. ABC expects that, if it enters the electricity market, it would be well placed to get the contract to supply that additional power.

The lead times required for ABC to build and commission its power stations are about ten years. This means that the plant configuration that it starts to build now must be matched to the electricity sector of ten years away, rather than the electricity sector that currently exists. ABC has decided that the principal driver of electricity use is economic activity. In addition, if economic activity is low, then XYZ is extremely unlikely to expand its operations, and thus increase its demand for power. If economic activity is high, then XYZ is likely to increase its production capacity, although it may choose to expand its operations in a different country. The other major driver that ABC have identified is the introduction of taxes on carbon dioxide emissions. Currently the government is giving very mixed messages about carbon taxes and ABC consider that the introduction of a tax within the next ten years is possible, but by no means certain. However, they do think that a carbon tax is very unlikely to be introduced if economic activity is low, both because carbon emissions would be curtailed naturally, and because difficult economic conditions would increase the political difficulties of introducing a carbon tax. A further complication is that it is unclear whether a carbon tax would increase, or decrease the demand for electricity.

ABC have built five scenarios to represent these drivers of the future. The first scenario is the "Low" scenario, under which economic activity is depressed, electricity demand grows slowly, and electricity prices are low. Under this scenario XYZ do not increase their demand for electricity, and no carbon taxes are introduced. Contrasting the Low scenario, are two scenarios "Dem1" and "Dem2", under which there is high economic activity, leading to rapid growth in electricity demand, and

high prices. These two scenarios are identical, except that under Dem2, XYZ increases its demand for electricity and ABC win the contract to supply it, whereas, under Dem1, XYZ either does not increase its demand, or ABC fails to win the contract.

The last two scenarios, "CO2A" and "CO2B", assume high economic activity followed by the introduction of a carbon tax. Two scenarios have been used to model the introduction of a carbon tax because it is unclear whether a tax would increase, or decrease, electricity demand. Under scenario, CO2A, demand for electricity is reduced to levels below those of Dem1 and Dem2, and the price is increased. Under CO2B, demand for electricity is increased to levels above those of Dem1 and Dem2, but the price is unaffected and remains the same as under Dem1 and Dem2. User XYZ does not increase its demand under either of the carbon tax scenarios.

Company ABC can start construction now, they can wait until things become clearer before starting, or they can start construction of some capacity now, and decide later whether to expand the construction programme. Their construction options are as follows. The section of river has three possible dam sites, each of which could be developed to create a reservoir, and install generating capacity. These sites can be developed in any combination. Construction can start now, or be deferred until later, again in any combination. ABC's engineering consultants have decided that there is only one suitable generating capacity for each site. The lead time to develop the hydro sites is ten years in all cases.

ABC can also build thermal generating plants, either gas-fired, or coal-fired. There are two options for coal-fired plant; either to burn black coal, or to burn lignite. Because of the geographical locations of the coal fields it is not possible for a single plant to burn both types of coal. ABC may choose to build more than one of each type of thermal plant, but because of constraints on the availability of fuel there are limits on the total capacity of each type of plant that can be built. The lead time to build and commission thermal generating stations is ten years. Once a thermal plant has been built it is possible for it to be expanded later. The lead time for the expansion of an existing thermal plant is five years.

ABC assume an economic life of thirty years for thermal plant, and thirty-five years for hydro stations. Both are assumed to have no residual value at the end of their economic lives.

The final option is to use wind generation. Wind generation is assumed to have

a five year lead time, and an economic life of twenty years. ABC do not consider that wind generation is a viable option at present, but they do think that they may wish to use it in the future.

ABC must make an immediate decision about what sort of plant (if any) to start building now. They then have opportunities to take recourse decisions later, in which case they either build additional plant at new sites, or expand completed thermal plant. Once a dam has been built it is not economically viable to expand capacity at that site.

8.3 The Formulation

The problem is formulated as a two-stage problem with a chance node in ten years' time, at which point it becomes apparent which scenario is occurring. To model the option of staged development, stage one is divided into two periods, each of five years, and construction of plant can begin at the beginning of either period. Stage two is also modelled as two periods, the first (period 3) being five years long, and the second (period 4) being thirty years long. Plant that is started in period 1 will be operational in period 3, and can also be expanded in period 3. Plant that was started in period 2 will not be available until period 4, but its construction can be modified in period 3 to increase its capacity. In both cases, the additional capacity will be available in period 4.

Construction of wind generation can be started at any of four available sites at the beginning of either of periods 2 or 3. The construction cost varies between the sites according to the difficulty of access and the distance from the electricity network. Because ABC do not consider that wind generation is a viable option at present it is not included in the period 1 alternatives.

The construction alternatives for stage one are modelled as binary variables. Each of the three hydro stations can be built in one size only, whereas there are several sizes of thermal plant that can be built. More than one thermal plant of a particular type can be built, but constraints on the availability of fuel limits the total capacity of each plant type. For gas this limit is 600 MW, for black coal 800 MW, and for lignite it is 1200 MW. A shortage of suitable wind generation sites limits the total wind capacity to 800 MW. The alternatives for building generating plant, and their costs, are shown in Table 8.1.

Plant Type	Size MW	Start in Period 1		Start in Period 2		Maximum Expansion MW
		Building Cost 10^5	Expansion Cost $/MW	Building Cost 10^5	Expansion Cost $/MW	
hydro site a	150	1030	—	1130	—	—
hydro site b	200	1530	—	1680	—	—
hydro site c	150	1100	—	1210	—	—
gas-fired a	100	555	3.7	610	3.3	100
gas-fired b	200	890	3.6	980	3.2	100
gas-fired c	300	1225	3.5	1350	3.5	150
gas-fired d	400	1560	3.2	1720	3.1	200
black coal a	200	980	4.2	1080	3.8	100
black coal b	400	1750	4.4	1930	4.0	200
black coal c	600	2530	4.5	2780	4.1	200
brown coal a	200	1080	4.4	1190	4.1	100
brown coal b	400	1920	4.8	2110	4.3	200
brown coal c	600	2750	5.2	3030	4.7	300
wind a	100	—	—	530	3.0	200
wind b	100	—	—	580	3.2	200
wind c	100	—	—	630	3.4	200
wind d	100	—	—	650	3.5	200

Table 8.1: List of Stage One Construction Options
Each plant type can be started in period 1 or period 2 of stage one. Only one option is available for each hydro site, but several options are available for each type of thermal plant. The cost of building the plant increases from period 1 to period 2. However, the cost of expansion in period 3 is less if the plant is started in period 2. This is because the expansion can be done as the plant is built, rather than after it has been completed.

As discussed in Section 7.5.1, it is often possible to prune the branch and bound tree by taking account of constraints that mean that only some combinations of values of the binary variables are feasible. This problem includes several such constraints and these are passed to the solution algorithm as branching rules to enable it to discard certain branches at the top of the tree. These rules state that because only one dam can be built at each site it is not possible to start construction at a site in both of periods 1 and 2. If we denote the three dam sites as a, b and c, then we have six binary variables, da1, da2, db1, db2, dc1 and dc2 representing the start of construction at each site in each of periods 1 and 2. By using these six variables

as the branching variables for the top six levels of the branch and bound tree the rules enable the algorithm to prune the tree as follows:

At level 1, variable da1 can be set to 0 or 1.

At level 2, variable da2 can be set to 0 or 1 if da1 = 0, but only to 0 if da1 = 1.

At level 3, variable db1 can be set to 0 or 1.

At level 4, variable db2 can be set to 0 or 1 if db1 = 0, but only to 0 if db1 = 1.

At level 5, variable dc1 can be set to 0 or 1.

At level 6, variable dc2 can be set to 0 or 1 if dc1 = 0, but only to 0 if dc1 = 1.

This enables the algorithm to prune off one branch at level 2, and so discard 1/4 of the tree. At level 4, 1/4 of the remaining 12 branches are pruned, and at level 6, 1/4 of the remaining 36 branches are pruned. The cumulative effect is that the algorithm can discard 27 of the 64 branches that would be in the tree at level 6, if it were not pruned. Clearly this provides a significant reduction in the computational effort required to analyse the tree and find the noninferior set.

The limits on the total capacity that can be built for each type of thermal plant provide an opportunity to further reduce the computational effort by checking to see if a proposed branching would violate these capacity constraints. This check is done before solving the sub-problem at each node. If the branching fails the check, then the algorithm fathoms the node immediately and backs up to the previous node. The check is formulated as a set of constraints. If the eight binary variables for the four gas-fired options in each of the first two periods are denoted as: ga1, ga2, gb1, gb2, gc1, gc2, gd1, and gd2, then the constraint is:

$$100ga1 + 100ga2 + 200gb1 + 200gb2 + 300gc1 + 300gc2 + 400gd1 + 400gd2 \leq 600$$

Similar constraints are formed for black coal and brown coal.

Programme listings of the model and the data, written using the "AMPL" modelling language, are included in Appendices B.1 and B.2.

8.4 Solving the Problem

This problem was solved using the multicriteria branch and bound algorithm developed in Chapter 7. The first pass through the branch and bound tree found the criterion vector $(540, 8387, 8631, 1078, 4651)$ shown in Table 8.2. The second pass then found the other criterion vectors listed in the table. The second pass found two

solutions before it found the maximal solution for the Low scenario. The maximal solutions for the remaining scenarios were found on the first pass through the tree for each scenario. These solutions show that coal is the preferred generating option in the absence of a carbon tax, with black coal used first and lignite added as the demand increases. However, neither solution builds plant to the maximum allowed capacity. When a carbon tax is introduced hydro becomes the preferred generating option, followed by gas as the demand increases.

Maximal for	Scenario Objective Value					Plant Built
Scenario	Low	Dem1	Dem2	CO2A	CO2B	MW
—	540	8,387	8,631	1,078	4,651	400 gas, 600 black
—	1,658	6,694	6,694	797	2,708	600 black
—	1,732	4,795	4,795	2,108	4,350	400 gas
Low	1,788	5,306	5,306	996	2,612	400 black
Dem1	-623	8,803	9,623	-1,400	2,096	600 black, 600 brown
Dem2	-2,543	8,548	10,133	-3,320	941	600 black, 1,000 brown
CO2A	748	4,434	4,434	3,739	5,897	500 hydro
CO2B	- 410	6,997	7,105	3,608	7,431	500 hydro, 400 gas

Table 8.2:
List of Noninferior Solutions Found by the First Two Passes Through the Branch and Bound Tree
The first solution was found during the pass one, which also determines the branch and bound tree. The remaining solutions were found by the second pass that finds the scenario-maximal solutions. In this example, two extra solutions were found before the scenario-maximal solution for scenario Low was found. The scenario-maximal solutions for the remaining scenarios were found on the first pass down the tree in each case.

It is apparent from Table 8.2 that several of the scenario-maximal solutions perform very badly under the other scenarios. In particular, choosing to build for the two high demand scenarios will lead to very bad outcomes under the scenarios Low and CO2A.

The third pass through the branch and bound tree finds the rest of the noninferior set, which is listed in Tables 8.3 and 8.4. These tables are sorted by installed capacity, and then by the objective function value of scenario Dem1. This problem has 63 noninferior solutions, and choosing among will be a non-trivial task. However, as observed in Chapter 7, the noninferior set of a mixed integer problem is a set of discrete points, and this makes analysis of the set much easier than is the case for continuous problems. Unlike the continuous case, the value path of a mixed integer

	Scenario Objective Function Values					Plant Built				Total
	Low	Dem1	Dem2	CO2A	CO2B	Hydro	Gas	Black	Brown	MW
1	1547	4437	4437	2802	4913	b1	b1			400
2	1585	4720	4720	2252	4053	b1		a1		400
3	1732	4795	4795	2108	4351		d1			400
4	1788	5306	5306	996	2612			b1		400
5	1055	4741	4741	3288	5574	b1 c1	a2			450
6	1074	4734	4734	3305	5595	b1 c1	a1			450
7	1525	4780	4780	2567	4844	c1	c1			450
8	761	4434	4434	3739	5897	a1 b1 c1				500
9	1483	5329	5329	3029	5484	b1	c1			500
10	1017	5315	5315	3452	5982	b1 c1	b1			550
11	1418	5591	5591	2686	5401	c1	d1			550
12	1367	6067	6067	3081	6021	b1	d1			600
13	1198	6265	6265	2574	5003	b1	b1	a1		600
14	1503	6559	6559	1806	4453		d1	a1		600
15	1358	6578	6578	1664	4224		d1		a1	600
16	1658	6694	6694	797	2708			c1		600
17	1352	6711	6712	2230	4292	b1		b1		600
18	1285	6957	6957	1086	3134		b1		b1	600
19	901	6030	6030	3498	6518	b1 c1	c1			650
20	1169	6305	6305	1968	4622	a1	c1	a1		650
21	685	6426	6426	2992	5677	b1 c1	a1	a1		650
22	1160	6510	6510	2226	4940	c1	c1	a1		650
23	1082	6943	6943	2623	5573	b1	c1	a1		700
24	946	6967	6967	2481	5347	b1	c1		a1	700
25	1015	7414	7414	2211	4797	b1	a1	b1		700
26	790	7436	7438	1982	4422	b1	a1		b1	700
27	571	6698	6704	3494	7006	b1 c1	d1			750
28	380	6892	6897	3014	6064	b1 c1	b1	a1		750
29	278	6914	6919	2875	5839	b1 c1	b1		a1	750
30	168	6916	6916	2753	5780	b1 c1	b1		a2	750
31	105	6966	6966	2736	6092	b1, c1	a1, a2	a1		750

Table 8.3: List of All Noninferior Solutions - part 1

The codes in the column headed "Plant Built" denote the site for hydro plant, the size alternative for thermal plant, and the period of construction. E.g. c1 in the Hydro column means built at site c in period 1, c2 in the Brown column means option c for brown coal, built in period 2.

	Scenario Objective Function Values					Plant Built				Total
	Low	Dem1	Dem2	CO2A	CO2B	Hydro	Gas	Black	Brown	MW
32	1010	6971	6972	1998	5183	a1	d1	a1		750
33	1001	7087	7091	2253	5496	c1	d1	a1		750
34	888	7129	7132	2113	5270	c1	d1		a1	750
35	515	7311	7359	2674	5312	b1 c1		b1		750
36	777	7395	7403	2630	6069	b1	d1	a1		800
37	665	7442	7449	2493	5874	b1	d1		a1	800
38	735	7733	7744	2266	5177	b1	b1	b1		800
39	861	7737	7781	1732	4388	b1		c1		800
40	558	7810	7818	2043	4812	b1	b1		b1	800
41	1271	7899	7905	1553	4615		d1	b1		800
42	1051	7983	7987	1305	4265		d1		b1	800
43	637	8067	8170	1403	3926	b1			c1	800
44	45	7410	7425	3011	6532	b1 c1	c1	a1		850
45	-57	7457	7471	2875	6356	b1 c1	c1		a1	850
46	495	7914	7921	1352	4423	a1	c1		b1	850
47	476	7988	7998	1672	4750	c1	c1		b1	850
48	-410	6997	7105	3608	7431	a1 b1 c1	d1			900
49	306	8161	8205	1627	4893	b1	a1	c1		900
50	223	8199	8246	2070	5382	b1	c1		b1	900
51	875	8270	8290	1078	4172		c1	c1		900
52	82	8377	8538	1323	4448	b1	a1		c1	900
53	318	8001	8231	1900	5519	c1	d1	b1		950
54	160	8096	8262	1370	4925	a1	d1		b1	950
55	141	8166	8337	1677	5245	c1	d1		b1	950
56	233	8189	8244	1290	4626	c1	b1	c1		950
57	65	8095	8490	2258	6018	b1	d1	b1		1000
58	-29	8257	8484	1635	5274	b1	b1	c1		1000
59	-112	8258	8597	2052	5747	b1	d1		b1	1000
60	540	8387	8631	1078	4651		d1	c1		1000
61	-102	8241	8621	1290	5066	c1	c1	c1		1050
62	-623	8803	9623	-1400	2096			c1	c1	1200
63	-2543	8548	10133	-3320	941			c1	b1,c1	1600

Table 8.4: List of All Noninferior Solutions - part 2
The codes in the column headed "Plant Built" denote the site for hydro plant, the size alternative for thermal plant, and the period of construction. E.g. c1 in the Hydro column means built at site c in period 1, c2 in the Brown column means option c for brown coal, built in period 2.

problem provides a complete representation of the available solutions, and of the trade-offs between them. The value paths for all 63 solutions for this problem are shown in Figure 8.1. The generating capacity built in stage one is superimposed on the same diagram to show how performance under the scenarios changes as the installed capacity changes. The solutions are in the same order as Tables 8.3 and 8.4, that is solution 1 at the left, and solution 63 at the right. The diagram shows that the objective function value for scenarios Low and CO2A gradually reduce as the generating capacity increases, while the values for scenarios Dem1 and Dem2 increase. However, when the capacity goes above 1,000 MW, although the results for scenarios Dem1 and Dem2 continue to increase, the results for the other scenarios drop dramatically, and at 1,600 MW reach catastrophic values. It is apparent that building more than 1,000 MW creates considerable overcapacity under all but the two scenarios with the highest demands. The diagram also shows that solutions that perform well under scenarios Dem1 and Dem2 perform poorly under the carbon tax scenarios. At each capacity level the high point for scenarios Dem1 and Dem2 generally corresponds to low points for the carbon tax scenarios. The range of values for the carbon tax scenarios at a particular generating capacity is much greater than for scenarios Dem1 and Dem2, suggesting that performance under the carbon tax scenarios is much more sensitive to the plant mix than it is under Dem1 and Dem2. This makes sense, because the differences in generating costs between the plant types are less than the size of the carbon tax. Scenario Low also moves against scenarios Dem1 and Dem2, and this movement is related to the size of the hydro plant. Hydro plant is expensive to build, and it is only under the scenarios with high demand, and/or a carbon tax, that the low unit generating cost produces sufficient benefit to recover the large initial investment.

It is also apparent that there is little difference in outcome between scenarios Dem1 and Dem2 until the installed capacity reaches 950 MW. This shows that the supply contract with Company XYZ only makes a difference to the company's returns if they build enough capacity to fill the contract in addition to supplying the rest of the market. Even where the contract is supplied in preference to the rest of the market the difference in revenue is not enough to make a significant difference between the objective function values of scenarios Dem1 and Dem2. This shows that it was unnecessary to build both scenarios. It would have been sufficient to have had a single, high demand scenario. The details of the pattern of demand, contract

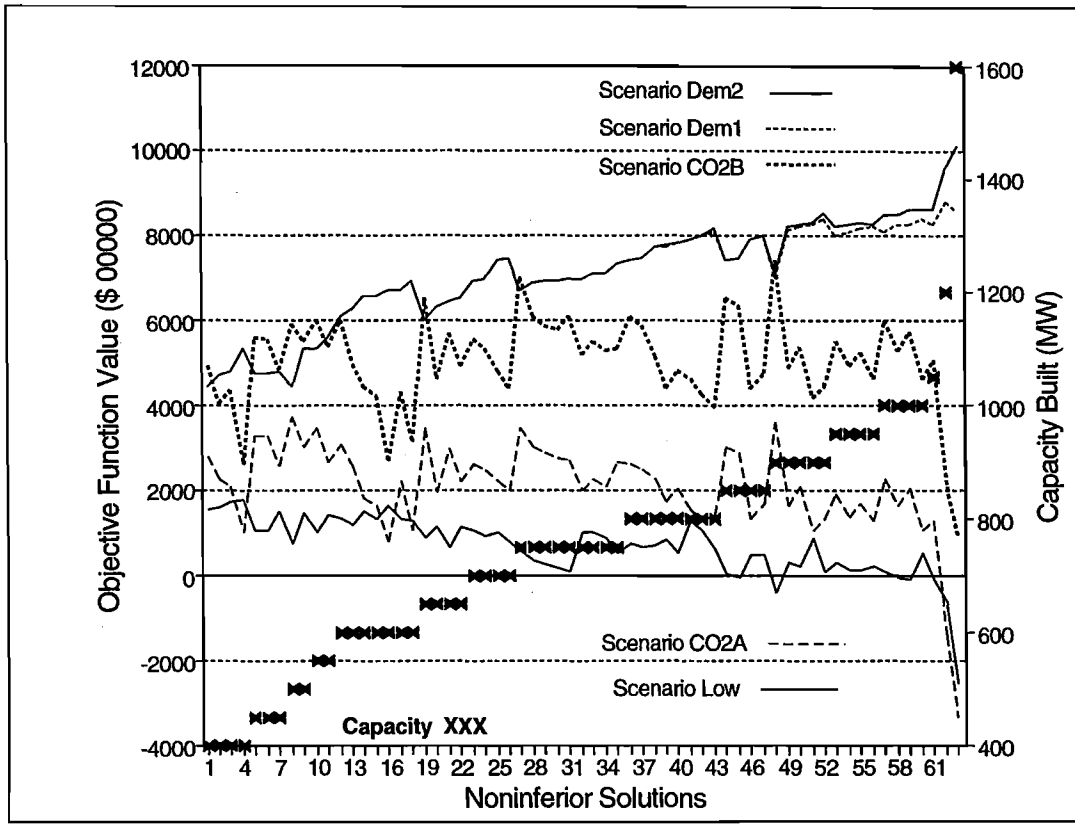

Figure 8.1: Value Path Showing All Noninferior Solutions
This diagram shows the value paths for the five scenarios, and the generating capacity to be installed by each solution. There are several solutions that build the same total capacity, but they include different mixes of plant. This means that there are trade-offs between the scenarios at each capacity level, based on the types of plant built.

or market, are not material to this problem.

Consideration of the full noninferior set provides considerable insight into the problem. However, 63 solutions are too many to consider when looking for a solution and the decision maker will want to start eliminating some of them in order to concentrate on those that could lead to an implementable decision. Some of the solutions are quite extreme, and the decision maker can be expected either to concentrate on them if they fit his/her requirements, or to eliminate them. In this case, a risk averse decision maker might decide to discard all solutions that lead to a loss under the Low scenario. Further solutions can be discarded by requiring that the objective function value be at least some minimum amount under every scenario. For example, if a minimum acceptable objective function value of 1,000 is used, then solutions 8, 16, 19, 21, 24, 26–31, 34–40, 43–63, will be discarded. The

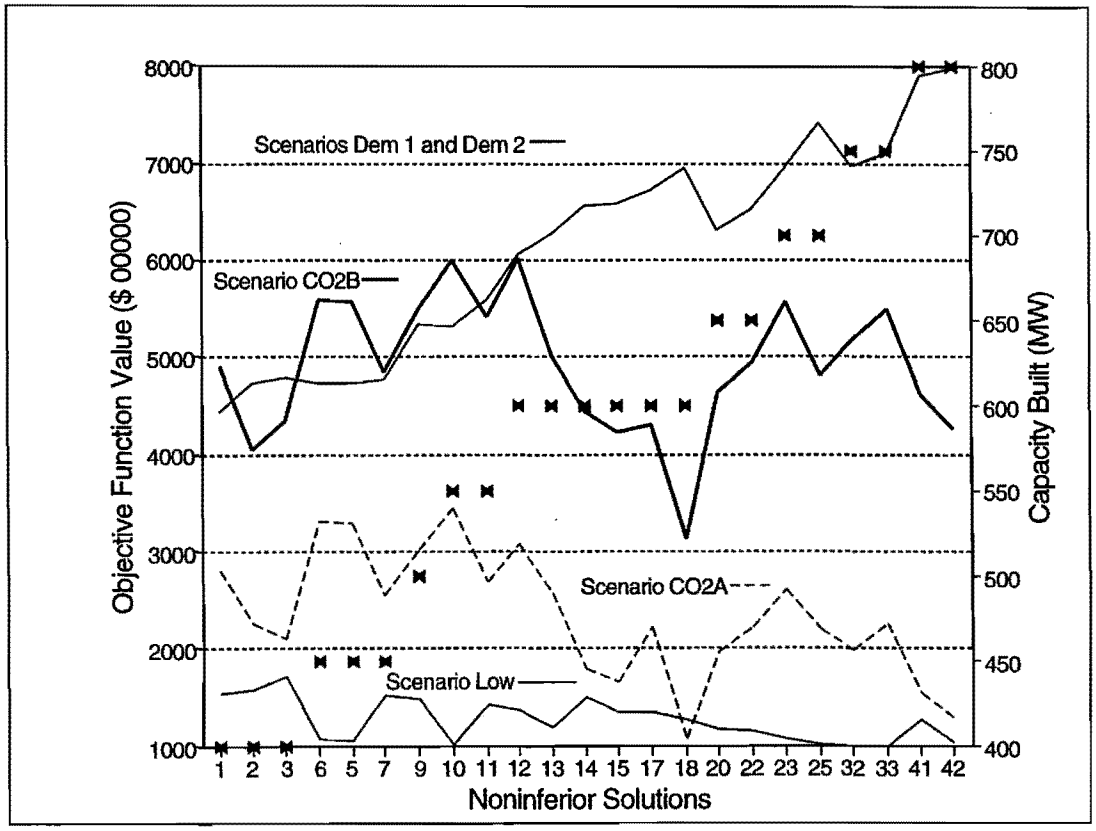

Figure 8.2:
Value Path For The Solutions That Achieve an Objective Function Value of at Least 1000
This is the value curve for the sub-set of solutions that produce an objective function value of at least 1000 for every scenario. Superimposed on the value curves is the total plant capacity. The trade-offs between the scenarios caused by change in plant type are clearly apparent for solutions 12 to 18, that build 600 MW. There is a shift from hydro and gas to black and then brown coal from 12 to 18, and the effects on the carbon tax and non-tax scenarios is quite apparent.

remaining solutions are shown in Figure 8.2. This figure shows that this requirement filters out all solutions that build more than 800 MW, and that it also eliminates several solutions from each capacity grouping. This diagram also shows the trade-off between the high demand scenarios and the carbon tax scenarios. However, some solutions trade-off the low demand scenario against the carbon tax scenarios, rather than the high demand scenarios. For example, the value path from solution 3 to solution 7 shows the carbon tax scenarios moving against the Low scenario, but the result under scenarios Dem1 and Dem2 is almost constant. Again, it is apparent that the trade-offs between the high demand scenarios and the carbon tax scenarios generally involve much larger changes for the carbon tax scenarios than for the high

demand scenarios. The most extreme case is the move from solution 12 to solution 18, which improves the result under scenario Dem1 by 1,000, but reduces the result under scenario CO2B by almost 3,000, and under CO2A by 2,000.

A somewhat different tack is to specify the filter as a requirement that the objective function value for every scenario must be at least some fraction of the scenario-maximal value of that scenario. Setting the requirement to be 50% of the scenario-maximal value reduces the list of candidate solutions to fourteen. That is, to solutions: 9, 10, 11, 12, 13, 17, 19, 20, 22, 23, 24, 25, 32 and 33. Increases in the required percentage further reduces the list, until, at 60%, only solutions 13 and 23 remain.

Figure 8.3 shows the fourteen solutions that remain when the 50% requirement is applied, and Figure 8.4 shows the corresponding building programmes. All of these solutions produce similar results under the Low scenario, and the main trade-offs are between the carbon tax scenarios and the high demand scenarios. The solutions that include cheap, but carbon intensive coal generation do well under Dem1 and Dem2, and poorly under the carbon tax scenarios, while the solutions that build the more expensive hydro plant behave in the opposite manner.

A different decision maker might choose to concentrate on the high demand scenarios and accept the risk of things turning out badly if a low demand future eventuates. Such a decision maker would be interested in a different subset of the noninferior solutions than that of the risk averse decision maker above. For example, such a decision maker might weed out all decisions that do not produce an objective function value of at least 6,000 under Dem1 and Dem2. A decision maker who was confident of observing high demands, but also wanted to hedge against a carbon tax would be drawn to solutions 12, 19, 27, and 48, which produce similar results under the three high demand scenarios, Dem1, Dem2 and CO2B. These solution stand out in Figure 8.1 as the solutions where the objective function values for Dem1 and Dem2 drop down, and the value for CO2B spikes up to meet them. The decision maker could then decide how much performance under the high demand scenarios should be traded off in order to hedge against a low demand future.

It is interesting to observe that several solutions pass several of these filtering criteria. Solutions 12 and 19 pass the requirement that all scenario objective function values reach at least 50% of the scenario-maximal value, they also reach 6,000 under the high demand scenarios, and they produce similar results whether, or not, a

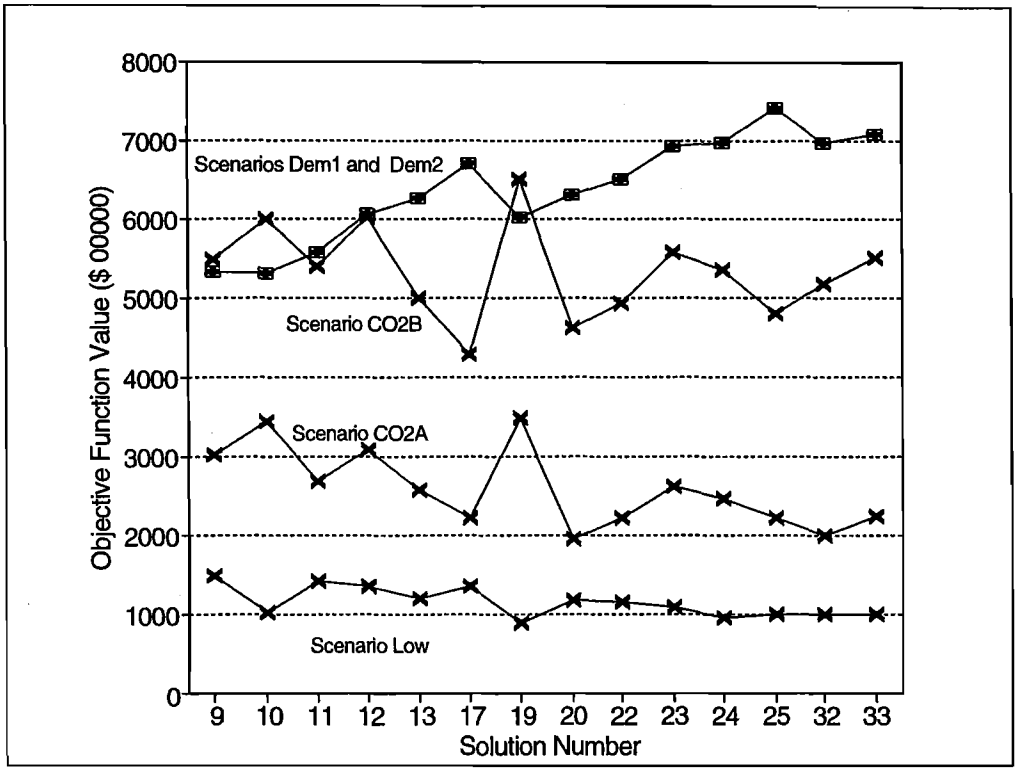

Figure 8.3:
Value Path for the Solutions That Achieve an Objective Function Value of at Least 50% of the Scenario-Maximal Values

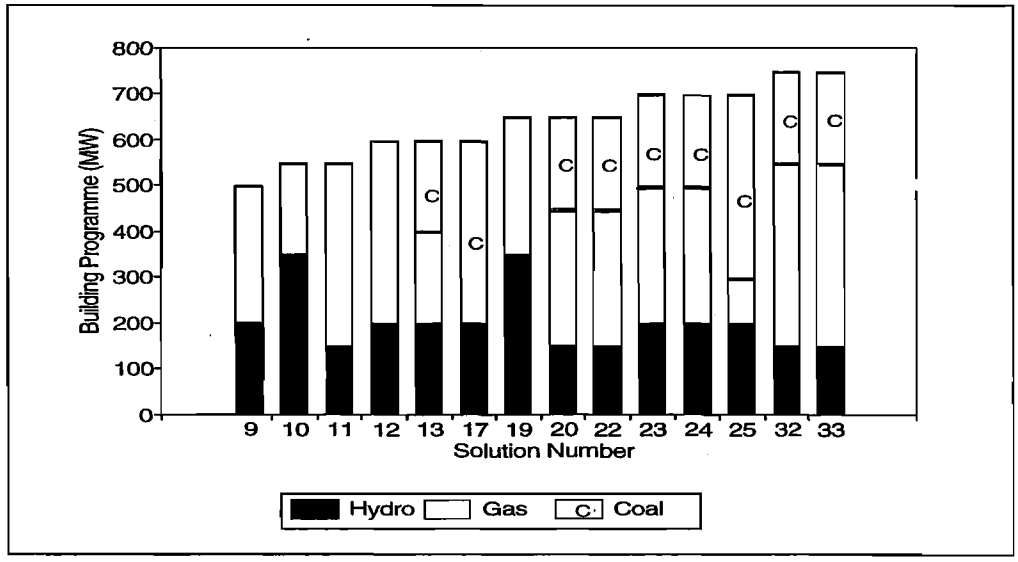

Figure 8.4:
Building Programme of the Solutions That Achieve at Least 50% of the Scenario-Maximal Objective Function Values

carbon tax is introduced. Solutions such as these will be of particular interest to decision makers because they will be acceptable to different stake holders who have different views of the problem. The more stake holders who find a proposed solution acceptable, the more likely it is that the decision makers will get approval for that decision, and that they will be able to implement it.

If the decision makers focus on solutions 12 and 19, they will observe that solution 19 fails to meet the requirement that the objective function value be at least 1,000 under all scenarios. Solutions 12 and 19 obtain the same result under scenarios Dem1 and Dem2, and the trade-off is between the carbon tax scenarios and the low demand scenario. The choice between these two solutions becomes a choice between hedging against a carbon tax future and a low demand future.

Many other patterns can be found, and decision makers would also obtain further insights by considering the recourse decisions of the efficient solutions. The recourse decisions have been included for the scenario-maximal solutions only, see Table 8.5, as an illustration of the additional information that is available from a set of efficient solutions. Similarly, Table 8.6 shows the energy demanded and supplied under each scenario for each of the scenario-maximal solutions. This information is, of course, available for all of the efficient solutions, although it has not been included here because of the space that would be required. Consideration of the recourse decisions associated with a solution gives the decision maker information about the implications of implementing that solution. For example, Table 8.5 shows that the scenario-maximal solution for the Low scenario builds a 400 MW coal-fired station burning black coal. The recourse decisions show that the plant would be expanded in stage two if the Low scenario occurs, and that an additional coal-fired plant would have to be built under the three high demand scenarios. All of the other scenarios also build 800 MW of wind generation at stage two. Table 8.6 shows that the scenario-maximal solution for the Low scenario leaves a great deal of demand unmet, and so misses out on a lot of potential revenue.

The objective of this chapter is not to analyse the example problem in detail, but to show the type of analysis that can be carried out, and the insights that are available when a set of efficient solutions is found for a problem. Far more insights are available from a set of efficient solutions than are available from a single solution that maximises a weighted objective function. For example, when this problem is solved using equal weights for the scenarios, solution 57 is found. Although

Maximal for Scenario	Plant Built Stage 1 MW	Scenario Observed	Plant Built Stage 2 MW	
Low	400 black	Low	98 black	
		Dem1	200 black	800 wind
		Dem2	200 black	800 wind
		CO2A		800 wind
		CO2B	186 black	800 wind
Dem1	600 black, 600 brown	Low		
		Dem1		188 brown
		Dem2	200 black	300 brown
		CO2A		800 wind
		CO2B		800 wind
Dem2	600 black, 1,000 brown	Low		
		Dem1		
		Dem2	146 black	200 brown
		CO2A		800 wind
		CO2B		800 wind
CO2A	500 hydro	Low		
		Dem1		800 wind
		Dem2		800 wind
		CO2A		724 wind
		CO2B		800 wind
CO2B	500 hydro, 400 gas	Low		
		Dem1	200 gas	200 wind
		Dem2	200 gas	249 wind
		CO2A		400 wind
		CO2B	200 gas	800 wind

Table 8.5:
The Stage One and the Stage Two (Recourse) Decisions of the Scenario-Maximal Solutions.
This table shows the plant to be constructed during stage one, and the plant to be built during stage two under each scenario.

Maximal for Scenario	Scenario Observed	Energy Demanded and Supplied per year (MW-weeks)					
		Period 3			Period 4		
		Demand	Supply	Short	Demand	Supply	Short
Low	Low	7,680	7,230	450	8,300	7,040	1,260
	Dem1	34,930	13,925	21,005	40,160	27,540	12,620
	Dem2	55,730	13,925	41,805	60,960	27,540	33,420
	CO2A	26,230	7,430	18,800	30,110	14,510	15,600
	CO2B	36,070	10,740	25,330	42,170	17,460	24,710
Dem1	Low	7,680	7,680	0	8,304	8,304	0
	Dem1	34,930	34,490	340	40,160	38,650	1,510
	Dem2	55,730	48,960	6,770	60,960	49,340	11,620
	CO2A	26,230	26,230	0	30,110	30,110	0
	CO2B	36,070	35,290	780	42,170	32,170	10,000
Dem2	Low	7,680	7,680	0	8,300	8,300	0
	Dem1	34,930	34,490	0	40,160	39,340	820
	Dem2	55,730	54,810	920	60,960	55,300	5,660
	CO2A	26,230	26,230	0	30,110	30,110	0
	CO2B	36,070	35,290	780	42,170	41,060	1,110
CO2A	Low	7,680	7,680	0	8,300	6,060	2,240
	Dem1	34,930	15,570	19,360	40,160	23,900	16,260
	Dem2	55,730	15,570	40,160	60,960	23,900	37,060
	CO2A	26,230	15,590	10,640	30,110	23,110	7,000
	CO2B	36,070	15,570	20,500	42,170	23,900	18,270
CO2B	Low	7,680	7,680	0	8,300	6,060	2,240
	Dem1	34,930	32,680	2,250	40,160	38,670	1,490
	Dem2	55,730	34,290	21,440	60,960	40,160	20,800
	CO2A	26,230	15,590	0	30,110	28,880	1,230
	CO2B	36,070	32,390	3,680	42,170	41,280	890

Table 8.6:
List of Scenario-Maximal Solutions, Showing the Energy Demanded and Supplied Under Each Scenario.
The energy is in MW-weeks. This is calculated for each demand mode as the product of the MW demanded (or supplied) the number of weeks in the period and the load factor. These are then summed over all demand modes.

this is an implementable solution, knowing this solution, but only this solution, provides a great deal less information about the problem, and the options available to the company, than the list of efficient solutions produced by our approach. It is interesting to note that solution 57 is well to the right of the noninferior set, and builds 1,000 MW of capacity. This is close to the region in Figure 8.1 in which the objective function values for three of the five scenarios drop dramatically. If demand turns out to be somewhat lower than the demand modelled in these scenarios, then this solution is likely to perform very badly. A decision maker who has access to Figure 8.1 can see this possibility, and may choose to carry out more analysis, or to select a solution further to the left. Perhaps a solution from the group that builds 950 MW. In contrast, decision makers who have only been told about solution 57 can only see that single point, and they have no other information to work with. They have no way of knowing about the "fall-off" to the right, or about the trade-offs involved in choosing to build less capacity as a hedge against a future of low electricity demand.

8.5 Summary

In this chapter we have applied the algorithm developed in Chapter 7 to an example strategic planning problem, in which a company must choose between various generating technologies in order to supply electricity to an electricity market in ten years' time. The size of that market, and the economic environment within which it will be operating are uncertain, and several alternative futures have been described by a set of scenarios. Some of the available technologies will perform much better under some scenarios than under others, but the company must decide which technologies to adopt before the future is known. An additional complication is that the company must decide how much capacity to build before the future size of the market is known.

In problems of this type it is not possible to determine probabilities for the scenarios, although it may be possible to rank them according to their relative likelihoods, and thus obtain relative weights. Alternatively, the decision maker could assign them relative weights according to their importance. Once weights have been assigned to the scenarios the problem can be solved to find the solution that maximises the weighted objective function value. However, it hardly seems

appropriate to make decisions of this magnitude on the basis of a single answer produced by a mathematical programme (or any other means). In the first instance the decision makers should be looking for insights into the problem. They need to gain an understanding of how the alternatives available to them can be expected to perform under different future conditions, and of the trade-offs involved in choosing between these alternatives. In order to choose between alternative courses of action they need to know what the alternatives are. They do, of course, know that they can build five different types of plant in various sizes and combinations, but they need guidance about which combinations are better than others, and under what circumstances. The decision makers will also want to avoid dominated decisions. For this problem there are several thousand feasible plant combinations that could be built, the vast majority of which are dominated, and the decision makers will need help in identifying nondominated ones to choose among.

The approach used here identified 63 efficient stage-one decisions for the decision makers to choose among. Even 63 alternatives are too many to consider all at once, but 63 is a great improvement on several thousand, and these alternatives are known to be non-dominated. These decisions can be presented to the decision makers in the form of graphs and tables which can be studied to gain insights about the problem and the available actions. Once some understanding of the problem and the available trade-offs has been gained the noninferior set can be reduced by applying additional criteria. In this example, the list of alternatives was reduced to fourteen by setting minimum acceptable values for the scenario objective function values.

Use of this example illustrates that a great deal more information about a strategic planning problem can be obtained by analysing the problem to find a set of efficient solutions, than is available from a single solution that maximises a weighted objective function value. This additional information comes at a greater computational cost than is required to find a single solution. However, the importance of strategic decision making problems such as this is certainly sufficient to justify this additional cost.

Chapter 9

Summary

9.1 Introduction

In this thesis we have presented a new application of mathematical programming to the problem of decision making under uncertainty. This approach enables us to relax three assumptions that underpin current applications. The first assumption is that probabilities can be obtained for the outcomes of the uncertain parameters, the second assumption is that the decision makers are risk neutral, and the third assumption is that all of the decision makers' concerns can be adequately represented in the model.

In many situations these assumptions are appropriate, and mathematical programming techniques can be used to obtain satisfactory decisions. These situations are typically operational in nature. That is, the problem is repeated on a regular basis, and the uncertain parameters can be expected to behave in much the same way at each repetition. This means that probability distributions can be determined to describe the behaviour of the uncertain parameters. The fact that the problem is repetitive also means that the decision makers can expect the results to converge on the expected value over time, which makes the assumption of risk neutral decision makers quite reasonable.

However, many decision making problems are not repetitive, and it is often impossible to obtain reliable probability distributions for the uncertain parameters. This is particularly true for strategic decision making problems, and the work presented here is designed to extend the applicability of mathematical programming from operational decision making under uncertainty to strategic decision making

under uncertainty.

In Section 9.2 we briefly review the characteristics of strategic decision making problems that make the application of mathematical programming difficult, and even inappropriate. In the Section 9.3 we summarise the work, and in Section 9.4 we consider special difficulties presented by strategic decision making problems and discuss how this work addresses these difficulties. Then, in Section 9.5 we consider the limitations NSSA, and discuss opportunities for future work. Finally, we make some concluding remarks in Section 9.6.

9.2 Strategic Decision Making

Generally, strategic decisions are important decisions because they involve commitment of significant resources, and they are difficult to reverse. Because the same situation will not occur again, a strategic decision making problem is one-off in nature, and a bad outcome on one occasion cannot be offset by a good outcome on another. (Similarly, a good outcome will not be offset by a bad outcome.) These characteristics of importance and non-repeatability mean that strategic decision makers cannot be assumed to be risk neutral.

Because many strategic decision making situations are at level 3 (Courtney, Kirkland & Viguerie 1997), it is impossible to describe all possible futures. The best that can be done is to develop representative scenarios that describe contrasting possible futures. It is impossible to develop scenarios that represent all of the ways in which the future may evolve, and it is not possible to assign probabilities to the scenarios. Although it is possible to assign the scenarios relative weights, and thus form a stochastic optimisation problem, it seems unlikely that many decision makers would be willing to implement the single solution that this approach will produce. In this type of poorly described situation decision makers need insights into, and understanding of, the problem. Achieving an understanding of the problem is much better served by the provision of a set of alternative courses of action, than it is by the provision of a single recommendation.

Another distinguishing characteristic of strategic planning is the importance of the decisions. Operating decisions are short-term in nature, and it is very unusual for a poor operating decision to threaten the survival of an organisation, although a sequence of poor operating decisions might do so. Operational decision making

usually includes rapid feedback of results, and the opportunity to make adjustments at low cost. On the other hand, a strategic decision that goes wrong can lead to disaster. Strategic decisions are long term in nature, they commit significant resources, and it can be difficult and expensive to make adjustments later. An organisation may have to wait several years before a strategic decision can be evaluated, by which time it may be too late to correct matters. The importance of strategic decisions, and the potential for disaster, mean that strategic decision makers cannot be assumed to be risk neutral. They are likely to be risk averse, although some will be risk takers.

Strategic decision makers must consider many issues when choosing an action to implement, and many of these issues cannot be expressed in mathematical programming terms. This means that the optimal solution to the mathematical programming problem is unlikely to be the optimal solution to the strategic decision making problem. But techniques such as stochastic optimisation provide just one solution. If this decision does not provide adequately for the non-quantifiable issues, then the decision makers are little further forward. Further, the decision makers will normally have to trade off between many criteria. When only one solution is presented by the solution method it is impossible for decision makers to balance the many aspects of the problem. They must either accept or reject the solution, or ask for a new one. When the non-quantifiable aspects of the problem are a significant part of the problem, the decision makers will be looking for insights to assist with identifying an implementable decision. Again, the decision makers are much better served by the provision of insights into the problem, and information about the responses that are available to them, than they are by the provision of a single recommendation.

9.3 Summary of This Work

The work presented here is designed to provide decision makers with a set of solutions to choose among, rather a single "optimal" solution. The uncertain future is summarised as a small number of scenarios, and these scenarios are viewed as being in competition. That is, a decision that prepares well for one scenario will prepare poorly for another scenario, and so the scenarios are "competing" for a good decision. The problem is formulated as a multiobjective optimisation problem, and an approximation to the noninferior set is found. This set of solutions is presented

to the decision makers who can choose from this set according to their attitudes to risk, and in response to the non-quantifiable aspects of the problem. This noninferior set is obtained without assigning probabilities to the scenarios, and so the assumption that probability distributions are available for the uncertain parameters can be relaxed. The noninferior set includes solutions for all possible assignments of probabilities to the scenarios, which means that no assumptions are made about the decision makers' attitudes to risk, and so the assumption of risk neutral decision makers is relaxed.

In the first part of this work we adapt a technique from the multiobjective optimisation literature (Solanki, Appino & Cohon 1993) and applied it to the problem of decision making under uncertainty. This technique finds an approximation to the noninferior set in criterion space, and thus identifies an approximation to the efficient set in solution space. In this context, a solution is said to be efficient if no other solution exists such that the objective under one scenario can be improved without making the objective under at least one other scenario worse.

We then discuss methods of presenting the noninferior set to decision makers, and how these presentations can be used by decision makers to select a decision to implement. We show how the selected noninferior criterion vector can be translated back into solution space to find the corresponding decision vector. We apply our approach to two examples from the stochastic programming literature and show how this approach is able to generate greater insights about the problem than are available from the stochastic optimisation solution. We also show that the stochastic optimal solutions found by the original formulations are made available to the decision makers as part of the noninferior set.

Strategic decision making problems often include choices that can be taken, or not taken, (e.g. to build, or not build, a new factory). Such alternatives must be modelled as binary variables, and the second part of the work extends our approach to problems that include binary variables. We limit our attention to problems in which the stage one decision variables are all binary, and the stage two variables are continuous. We develop a branch and bound algorithm to find an approximation to the noninferior set for this type of problem, and we apply this algorithm to an example problem. In this problem a company is faced with a capacity expansion problem with a thirty year horizon. There are several thousand feasible combinations of the stage one binary variables, and the company must choose one to implement.

Our branch and bound algorithm identifies 63 plant combinations that are non-dominated. We show how these solutions can be presented to the decision makers to produce useful insights about the problem and the choices available to them. We then show how the decision makers can apply additional criteria to filter out much of the noninferior set and so focus on a small number of decisions to choose among.

9.4 Further Discussion

There is an important difference between decision making under uncertainty, and the deterministic, multiobjective optimisation problem that noninferior set scenario analysis is based on. In decision making under uncertainty only one scenario will actually occur, and the decision maker will observe one outcome, and one objective function value. In contrast, in deterministic multiobjective decision making, the decision maker will observe every objective function value. This means that the decision maker gets a "whole package", and a poor value for one objective can be compensated for by a good value for another. When decisions are taken under uncertainty, a poor result under the observed scenario is all that the decision maker gets. Knowledge that a better result would have been obtained had a different scenario occurred is little compensation. Further, the decision maker will feel regret if a different decision that would have performed better under the observed scenario was available, but not implemented. This will be especially true if the decision maker gambled on the observed scenario not occurring and traded off against it for a good result under a scenario that did not eventuate. On the other hand, of course, a decision maker may gamble and win. Decision making under uncertainty involves considerations of risk, and attitudes to risk, that are not part of deterministic multiobjective decision making.

A challenge for decision makers confronted by uncertainty is the need to determine what could happen without being influenced by their desires for what should happen. There is a danger that the scenarios developed to describe the future will be tailored to match the decision makers' desires about how they want the future to turn out. This will tend to remove uncomfortable futures from the representation of the problem. This behaviour may also influence the choice of scenario probabilities. An unattractive scenario can be "down-graded" by assigning it a lower probability than would be assigned by a truly objective decision maker.

The danger that decision makers will develop scenarios that reflect their preferences about what should happen, rather than an objective belief about what could happen, is a problem for all decision making tools that use scenarios to represent the uncertainty. This danger can only be guarded against by ensuring that people with contrasting world views are employed to develop the scenarios, and that the scenarios are critically reviewed.

The danger that decision makers may assign scenario probabilities that reflect preference rather than belief is a particular problem for stochastic optimisation, because the bias is applied before the problem is solved, and the optimal solution will be driven to favour the preferred scenarios. The bias becomes part of the formulation, and (unless it is extreme) will not be apparent when viewing the solution. In NSSA, this bias comes into play after the problem is solved, when the decision makers' choose their preferred solution. The decision makers may choose a solution that favours the scenario that they hope will happen, because they hope it will happen, and not because of its relative importance. This behaviour could lead the decision makers to choose a solution that corresponds to a higher level of risk taking than is consistent with their attitudes to risk.

However, NSSA does provide some insight into the issue that is not available with stochastic optimisation. The bias created by the choice of probabilities is buried in the problem formulation of the stochastic optimisation problem. On the other hand, because NSSA presents a set of solutions to choose among, the application of bias to the choice of solution appears as a "movement" across the noninferior set that may be recognised as such. Further, the selected solution can be found in weighting space and the corresponding scenario weights identified. If one scenario has a very large weighting, then this suggests that the choice of solution may have been biased by the decision makers' desires about the future, and that it should be reviewed.

In our view, decision makers can be expected to bring criteria to the problem that cannot be included in the mathematical programming formulation. Although these concerns cannot be included in the formulation, the decision maker is unlikely to ignore them. Instead, these criteria will be applied to the solution (or solutions) produced by the mathematical programme. The criteria applied to the solution after the event may be in conflict with the criteria used in the model, and thus any subsequent adjustments may not make sense in terms of the original mathematical formulation. For example, a model might be used to maximise an objective over

an organisation as a whole. But the general managers of the divisions of the organisation then evaluate the solution using other criteria, such as "fairness", and press for adjustments to the solution to meet these additional criteria. In this example, the concern about "fairness" was not part of the formulation, and may well be impossible to formulate. It may make little sense in the context of the mathematical formulation, and adjustments outside the model to increase "fairness" could lead to a solution that is seriously sub-optimal, or even infeasible, in terms of the mathematical model.

Strategic decisions are often associated with redistributions, as costs and benefits are seldom evenly distributed. For example, expansion of generating capacity may benefit all customers in a wide area, but only those living in the vicinity suffer from the resulting pollution. Such issues should be considered by decision makers, but the trade-offs are often very difficult to quantify. In this example, how should reductions in generating costs for all customers be traded off against increases in pollution around the power plant? Because NSSA produces a set of decisions to choose among, this type of trade-off can be considered after the problem is solved rather than having to be done as part of the formulation. An example of this type of issue is discussed in Section 6.3.1, in which choosing between different types of generating plant changes the impact on a local community. In the example, the decision maker is able to consider trade-offs between the objective of minimising generating cost, and the concerns of the local community.

Model formulations that produce a single, optimal solution are very vulnerable to the application of additional criteria after the model has been solved. No guidance is available to the decision makers to help them determine the effects of any adjustments they may make, although they can test adjusted solutions in the model. In contrast, because NSSA provides a set of alternative solutions, decision makers can include additional criteria in their choice of solution. This means that they can evaluate the trade-offs between the additional criteria, and the criteria included in the model. They also know (provided they keep within the noninferior set) that they have a solution that is both non-dominated, and feasible in the mathematical formulation.

When planning over long horizons, the criteria by which outcomes are evaluated may change over time. Either because society's world view has changed, or because the stakeholders have changed. Decision makers may wish to include this possibility

in their formulation of the problem, and one way of doing this is to create scenarios with objective functions that reflect these different criteria. This may lead to the objective functions of the different scenarios being in different units. For the problem to be formulated as a stochastic optimisation problem, all of the objective functions would have to be brought to the same units. Because NSSA does not form a composite objective function, the scenario objectives can remain in different units. This avoids the need to determine the value of one unit in terms of another before the problem can be solved. Such *a priori* valuations represent tradeoffs between the units, and thus tradeoffs between the scenarios. Because the NSSA formulation keeps the objectives in their native units, these tradeoffs will be considered as part of the process of choosing a decision, rather than being determined beforehand and becoming part of the formulation. As with the scenario probabilities, it is a question of visibility. In stochastic optimisation the tradeoffs implied by converting between units become part of the formulation, and disappear from sight. In NSSA, because these tradeoffs form part of the choice of final solution, they are kept in view, and can be challenged.

Scenario modelling may be viewed as a means of influencing how decision makers view the world (e.g. Wack 1985*a*, Wack 1985*b*), or as a tool for deciding on a course of action, as in stochastic optimisation. These two views overlap, in that it would be unusual for a decision maker to be interested in scenario models of the future without having a related decision making problem. It is also likely that the world view of a decision maker who wants to decide on a course of action will be influenced by insights generated by the process of deciding that course of action. The work presented here is directed towards deciding on a course of action, although it is also able to provide rich insights about the problem situation. In the work reported by Wack, contrasting scenarios were prepared with the intention of showing that the futures represented by some of them were implausible. The object was to convince the managers at Shell that their current view of the future should be abandoned, it was not to present a set of possible futures that should be accounted for in planning. NSSA, on the other hand, is designed to identify solutions that do account for all of the possible futures represented by the scenarios. The scenarios prepared by Wack would be precursors to the preparation of scenarios to be used as inputs into a decision making process aimed at deciding on a course of action. The scenarios included in the search for a course of action would not include the implausible

scenarios. These would have been discarded at an earlier stage in the process.

There is a characteristic of NSSA, however, that may soften more extreme world views of the decision makers and lead to the choice of less extreme decisions. NSSA presents many solutions to choose among, and it shows the tradeoffs that are being made between the scenarios as the decision makers move between solutions. This may tend to move decision makers away from extreme solutions, and towards solutions of compromise. In many situations, getting the best possible result for a particular scenario imposes a very poor result on other scenarios. Typically, the slope of the tradeoff surface is very steep (or flat) close to the extreme solutions, and significant improvements for the poorly performing scenarios can be achieved at a small cost to the favoured scenario. This was the case in the example in Chapter 8, where the solutions that performed best under the high demand scenarios performed very poorly under the other scenarios. The decision makers were able to significantly improve the situation for the other scenarios at small cost to the high demand ones. The fact that the decision makers can see these tradeoffs may encourage more moderate behaviour and increase their willingness to look for compromise.

9.5 Limitations of NSSA and Future Work

As presented in this thesis, NSSA has been limited to problems with two or three scenarios. This is in keeping with the recommendations of many authors in the planning literature (e.g. Schnaars 1987, Wack 1985*a*), who suggest that decision makers find more that three scenarios unmanageable. However, this view is not universally shared. For example, Mobasheri, Orren & Sioshansi (1989) use twelve scenarios in their work at Southern California Edison. It may well be that the recommendation that only three scenarios should be used will be modified in response to the greatly improved multi-media capabilities of modern computers. The mathematics of the XNISE algorithm used in this work apply to higher dimensions, and the approach could be extended to problems with more than three scenarios.

A more fundamental limitation of this work, however, is the presentation of the results. The techniques used here are static, two-dimensional presentations that can be printed on paper. However, there are many multi-media programmes that could be exploited for this purpose. These programmes would enable the decision maker to work with the noninferior set interactively in ways that are not described in this

work, although an interactive method for reducing the noninferior set is presented in Section 5.4.5. To use NSSA effectively, decision makers will need tools with which to explore the results, both to derive insights, and to choose a final solution to implement. Techniques developed for the multicriteria problem could be adapted to this problem. For example, there are the Feasible Goals Method (FDM) and Interactive Decision Maps (IDM) techniques developed by Lotov and others (Lotov, Chernykh, Bushenkov, Wallenius & Wallenius 1997, Lotov 1995).

Our work conceptualises scenarios as multiple objectives. It is limited to the two-stage problem in which the scenarios branch off from the present at a single point in time. The obvious next step is to extend this idea to multi-stage problems in which the event tree has branches at several points in time. When we view the path from the present out to the end of each branch as being a scenario, then groups of scenarios will share common pasts, and the decisions taken in these common pasts must be the same under each scenario in the group. This need to impose nonanticipativity on the decisions is, of course, the reason for formulating the problem with multiple stages. The scenario aggregation approach (Rockafellar & Wets 1991) formulates multi-stage scenarios as separate paths into the future, and then progressively enforces the nonanticipativity requirements. It would seem that there may be potential for applying this approach to the multi-stage NSSA problem. Whatever approach is used, however, the multistage problem will become very large, and impose serious computational demands.

NSSA formulates the scenario planning problem as a multiobjective problem, in which each scenario is viewed as having a competing objective. It is quite conceivable that the decision makers might have multiple objectives under each scenario, and that some of these objectives might be different under different scenarios. This would lead to a formulation in which each scenario contains multiple, competing objectives. The decision maker then has to trade off between objectives within the scenarios, as well as between scenarios. For example, a problem might have three scenarios, each of which has two objectives. The stage two problem under each scenario will also be a multiobjective problem. This means that for each stage one solution there will be a trade-off frontier under each scenario. For decision makers to compare two stage one decisions they will have to solve the multiobjective problem at stage two for each scenario and choose the recourse decision that they would implement if the scenario occurred. Having done that, the problem becomes one of choosing between

solutions by comparing three pairs of outcomes for each solution, as illustrated in Table 9.1.

	Scenario 1		Scenario 2		Scenario 3	
	Objective		Objective		Objective	
	A	B	A	B	A	B
Solution 1	300	80	150	150	180	120
Solution 2	420	60	200	80	120	200

Table 9.1:
Outcomes for a Problem with Two Objectives Under Each Scenario.

9.6 Concluding Remarks

In this work we have attempted to extend the application of mathematical programming to a wider range of decision making problems that include uncertainty. We have developed an approach that can be applied to problems that cannot be described in enough detail for stochastic optimisation to be an appropriate solution method. We have also tried to move away from generally accepted assumptions that it is sufficient (and appropriate) to optimise the expected value of the outcomes and to present a single solution vector to the decision makers. Instead, we have developed an approach that presents alternative, good quality solutions to decision makers. The decision makers can use these solutions to gain insights about their problem, and they can apply criteria to their choice of decision that cannot be included in a mathematical programming formulation. We hope that this work will contribute to a shift of emphasis away from the provision of answers towards providing insights about, and understanding of, the problem situation and the alternatives available to the decision makers.

Acknowledgements

My thanks to my supervisors, Dr John George and Dr Grant Read, for their support, advice and patience.

I wish to acknowledge the generous financial support I received from ECNZ, and to thank Tom Halliburton for organising it.

I also wish to thank the Energy Modelling Research Group of the Department of Management at the University of Canterbury for paying my fees.

And last but not least, a warm thank you to the staff, and the other PhD students, of the Department of Management for your friendship over the years it took to get this done. Mostly I enjoyed it, but I'm glad it's finished.

Bibliography

Bellman, R. E. (1957), *Dynamic Programming*, Princeton University Press.

Bellman, R. & Zadeh, L. (1970), 'Decision-making in a fuzzy environment', *Management Science* **17**(4), B–141–B–164.

Berland, N. J. & Haugen, K. K. (1996), *Mixing stochastic dynamic programming and scenario aggregation*, Vol. 64 of *Annals of Operations Research*, Scientific Publishing Company, pp. 1–19.

Birge, J. (1988), An l-shaped method computer code for multi-stage stochastic linear programs, *in* Y. Ermoliev & R. J.-B. Wets, eds, 'Numerical Techniques for Stochastic Optimization', Springer-Verlag, New York, chapter 12.

Birge, J. & Louveaux, F. (1988), 'A multicut algorithm for two-stage stochastic linear programs', *European Journal of Operations Research* **34**, 384–392.

Birge, J. R. (1982), 'The value of the stochastic solution in stochastic linear programs with fixed recourse', *Mathematical Programming* **24**, 314–325.

Birge, J. R. (1985), 'Decomposition and partitioning methods for multi-stage stochastic linear programs', *Operations Research* **33**(5), 989–1007.

Birge, J. R. (1995), *Models and Model Value in Stochastic Programming*, Vol. 59 of *Annals of Operations Research*, Scientific Publishing Company, pp. 1–18.

Birge, J. R. (1997), 'Stochastic programming computation and applications', *INFORMS Journal on Computing* **9**(2), 111–133.

Birge, J. R. & Louveaux, F. (1997), *Introduction to Stochastic Programming*, Springer Series in Operations Research, Springer-Verlag, New York.

Birge, J. R. & Rosa, C. H. (1995), 'Modeling investment uncertainty in the costs of global co2 emission policy', *European Journal of Operational Research* **83**, 466–488.

Bitran, G. R. G. R. (1977*a*), 'Linear multiple objective programs with zero-one variables', *Mathematical Programming* **13**, 121–139.

Bitran, G. R. G. R. (1977*b*), 'Theory and algorithms for linear multiple objective programs with zero-one variables', *Mathematical Programming* **17**, 362–390.

Carino, D. R., Kent, T., Myers, D. H., Stacy, C., Sylvanus, M., Turner, A. L., Watanabe, K. & Ziemba, W. T. (1994), 'The russell-yasuda kasai model: An asset/liability model for a japanese insurance company using multi-stage stochastic programming', *Interfaces* **24**(1), 29–49.

Chanas, S. & Kuchta, D. (1996), 'Multiobjective programming in optimization of interval objective functions - a generalized approach', *European Journal of Operational Research* **94**, 594–598.

Chankong, V. & Haimes, Y. Y. (1983), *Multiobjective Decision Making*, Vol. 8 of *North Holland series in system science and engineering*, Elsevier Science Publishing Co.

Charnes, A. & Cooper, W. W. (1963), 'Deterministic equivalents for optimizing and satisficing under chance constraints', *Operations Research* **11**(1), 18–39.

Charnes, A., Cooper, W. W. & Symonds, G. H. (1958), 'Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil production', *Management Science* **4**(3).

Chun, B. J. & Robinson, S. M. (1995), *Scenario analysis via bundle decomposition*, Vol. 56 of *Annals of Operations Research*, Scientific Publishing Company, pp. 39–63.

Cohon, J. L. (1978), *Multiobjective Programming and Planning*, Academic Press, New York.

Cohon, J. L., Church, R. L. & Sheer, D. P. (1979), 'Generating multiobjective trade-offs: An algorithm for bicriterion problems', *Water Resources Research* **15**(5), 1001–1009.

Courtney, H., Kirkland, J. & Viguerie, P. (1997), 'Strategy under uncertainty', *Harvard Business Review* pp. 67–79.

Dai, D., Carpenter, T. & Mulvey, J. (1997), 'Making a case for robust optimization models', *Management Science* **43**(7), 895–907.

Dantzig, G. B. & Glynn, P. W. (1990), *Parallel Processors for Planning Under Uncertainty*, Vol. 22 of *Annals of Operations Research*, Scientific Publishing Company, pp. 1–21.

Dantzig, G. B. & Infanger, G. (1993), *Multi-stage stochastic linear programs for portfolio optimization*, Vol. 45 of *Annals of Operations Research*, Scientific Publishing Company, pp. 59–76.

Dapkus, W. D. & Bowe, T. R. (1984), 'Planning for new electric generation technologies: a stochastic dynamic programming approach', *IEEE Transactions on Power Apparatus and Systems* **PAS-103**(6), 1447–1453.

Dembo, R. S. (1991), *Scenario Optimization*, Vol. 30 of *Annals of Operations Research*, Scientific Publishing Company, pp. 63–80.

Dembo, R. S. (1993), *Scenario immunization*, Cambridge University Press, chapter 12.

Dembo, R. S., Chiarri, A., Martin, J. G. & Paradinas, L. (1990), 'Managing hidroeléctrica española's hyrodelectric power system', *Interfaces* **20**(1), 115–135.

Enzer, S. & Alter, S. (1978), 'Cross-impact analysis and classical probability: The question of consistency.', *Futures* **10**(3), 227–239.

Eppen, G. D., Martin, R. K. & Schrage, L. (1989), 'A scenario approach to capacity planning', *Operations Research* **37**(4), 517–527.

Gassmann, H. I. (1989), 'Optimal harvest of a forest in the presence of uncertainty', *Canadian Journal of Forest Research* **19**, 1267–1274.

Gassmann, H. I. (1990), 'Mslip: A computer code for the multistage stochastic linear programming problem', *Mathematical Programming* **47**, 407–423.

Gassmann, H. I. & Wallace, S. W. (1996), *Solving linear programs with multiple right-hand sides: Pricing and ordering schemes*, Vol. 64 of *Annals of Operations Research*, Scientific Publishing Company, pp. 237–259.

Geoffrion, A. M. (1976), 'The purpose of mathematical programming is insight, not numbers.', *Interfaces* **7**(1), 81–92.

Gero, M. B. J. S. (1985), 'The noninferior set estimation (nise) method for three objective problems', *Engineering Optimization* **9**, 77–88.

Guldmann, J.-M. (1983), 'Supply, storage and service reliability decisions by gas distribution utilities: A chance-constrained approach', *Management Science* **29**(8), 884–906.

Gutierrez, G. J. & Kouvelis, P. (1995), *A robustness approach to international sourcing*, Vol. 59 of *Annals of Operations Research*, Scientific Publishing Company, pp. 165–193.

Haugen, K. K. (1996), 'A stochastic dynamic programming model for scheduling of offshore petroleum fields with recourse uncertainty', *European Journal of Operational Research* **88**(1), 88–100.

Haugland, D. & Wallace, S. W. (1988), 'Solving many linear programs that differ only in their righthand side', *European Journal of Operational Research* **37**, 318–324.

Helmer, O. (1981), 'Reassessment of cross-impact analysis', *Futures* **13**(5), 389–400.

Hof, J. G., Kent, B. M. & Pickens, J. B. (1992), 'Chance constraints and chance maximization with random yield coefficients in renewable resource optimization', *Forest Science* **38**(2), 305–323.

Hof, J. G. & Pickens, J. B. (1991), 'Chance-constrained and chance-maximizing mathematical programs in renewable resource management', *Forest Science* **37**(1), 308–325.

Huang, G. H., Baetz, B. W. & Patry, G. G. (1995), 'Grey integer programming: An application to waste management planning under uncertainty', *European Journal of Operational Research* **83**, 594–620.

Huang, G. & Moore, R. D. (1993), 'Grey linear programming, its solving approach, and its application', *International Journal of Systems Science* **24**(1), 159–172.

Infanger, G. (1992), *Monte Carlo (Importance) Sampling Within a Benders Decomposition Algorithm for Stochastic Linear Programs*, Vol. 39 of *Annals of Operations Research*, Scientific Publishing Company, pp. 69–95.

Infanger, G. (1994), *PLANNING UNDER UNCERTAINTY Solving Large-Scale Stochastic Linear Programs*, The Scientific Press Series, South-Western Publishing Co.

Inuiguchi, M. & Sakawa, M. (1995), 'Minimax regret solution to linear programming problems with an interval objective function', *European Journal of Operational Research* **86**, 526–536.

Isermann, H. (1974), 'Proper efficiency and the linear vector maximum problem', *Operations Research* **22**(1), 189–191.

Ishibuchi, H. & Tanaka, H. (1990), 'Multiobjective programming in optimization of the interval objective function', *European Journal of Operational Research* **48**, 219–225.

J.C., J. C. D. & Godet, M. (1975), 'Smic 74 - a method for constructing and ranking scenarios', *Futures* **7**(4), 302–312.

Jönsson, H., Jörnsten, K. & Silver, E. A. (1993), 'Application of the scenario aggregation approach to a two-stage, stochastic, common component, inventory problem with a budget constraint', *European Journal of Operational Research* **68**, 196–211.

Jörnsten, K. O. (1992), 'Sequencing offshore oil and gas fields under uncertainty', *European Journal of Operational Research* **58**, 191–201.

Julong, D. (1987), Grey decision making and its use for the determination of irrigation strategies, *in* J. Kacprzyk & S. Orlovski, eds, 'Optimization Models Using Fuzzy Sets and Possibility Theory', Reidel, Dordrecht, pp. 447–458.

Kall, P. & Wallace, S. W. (1994), *Stochastic Programming*, John Wiley & Sons Ltd., Chichester England.

Karaivanova, J. N., Narula, S. C. & Vassilev, V. (1993), 'An interactive procedure for multiple objective integer linear programming problems', *European Journal of Operational Research* **68**, 344–351.

Kelly, P. (1981), 'Further comments on cross-impact analysis', *Futures* **8**, 341–345.

Kiziltan, G. & Yucaoğlu, E. (1983), 'An algorithm for multiobjective zero-one linear programming', *Management Science* **29**(12), 1444–1453.

Klein, D. & Hannan, E. (1982), 'An algorithm for the multiple objective integer linear programming problem', *European Journal of Operational Research* **9**, 378–385.

Kluyver, C. A. D. & Moskowitz, H. (1984), 'Assessing scenario probabilities via interactive goal programming', *Management Science* **30**(3), 273–278.

Lotov, A., Chernykh, O., Bushenkov, V., Wallenius, H. & Wallenius, J. (1997), 'Interactive decision maps, with an example illustrating ocean waste management decisions', Paper presented at the Conference of the International Society for Multiple Criteria Decision Making, held in Cape Town, South Africa. Copy obtained from Web Site: http://www.ccas.ru/mmes/mmeda/.

Lotov, A. V. (1995), 'Computer-based support for planning and negotiation on environmental rehabilitation of water resource systems', Paper presented at the NATO ARW on ENVIRONMENTAL REHABILITATION OF WATER RESOURCE SYSTEMS in Yaroslavl, Russia. copy at Web Site: http://www.ccas.ru/mmes/mmeda/.

Louveaux, F. & Smeers, Y. (1988), Optimal investments for electricity generation: A stochastic model and a test-problem, *in* Y. Ermoliev & R. J.-B. Wets, eds, 'Numerical Techniques for Stochastic Optimization', Springer-Verlag, New York, chapter 24.

Marcotte, O. & Soland, R. M. (1986), 'An interactive branch-and-bound algorithm for multiple criteria optimization', *Management Science* **32**(1), 61–75.

Mavrotas, G., Diakoulaki, D. & Papayannakis, L. (1999), 'An energy planning approach based on mixed 0-1 multiple objective linear programming', *International Transactions in Operational Research* **6**, 231–244.

McLean, M. (1981), 'Does cross-impact analysis have a future?', *Futures* **8**, 345–349.

Miller, B. L. & Wagner, H. M. (1965), 'Chance constrained programming with joint constraints', *The Journal of the Operations Research Society of America* **13**(6), 930–945.

Mobasheri, F., Orren, L. H. & Sioshansi, F. P. (1989), 'Scenario planning at southern california edison', *Interfaces* **19**(5), 31–44.

Mulvey, J. M., Vanderbei, R. J. & Zenios, S. A. (1995), 'Robust optimization of large-scale systems', *Operations Research* **43**(2), 264–281.

Parkan, C. (1994), 'Decision making under partial probability information', *European Journal of Operational Research* **79**, 115–122.

Pasternak, H. & Passy, U. (1973), *Bicriterion Mathematical Programming with Boolean Variables*, University of South Carolina Press, Colombia, pp. 327–348.

Pereira, M. & Pinto, L. (1985), 'Stochastic optimization of a multireservoir hydroelectric system: A decomposition approach', *Water Resources Research* **21**(6), 779–792.

Pereira, M. V. F. (1989), 'Optimal stochastic operations scheduling of large hydroelectric systems', *Electrical Power and Energy Systems* **11**(3), 161–169.

Ramesh, R., Zionts, S. & Karwan, M. H. (1986), 'A class of practical interactive branch and bound algorithms for multicriteria integer programming', *European Journal of Operational Research* **26**, 161–172.

Read, E. G. & Boshier, J. F. (1989), Biases in stochastic reservoir scheduling models, *in* A. O. Esogbue, ed., 'Dynamic Programming for Optimal Water Resources Systems Analysis', Prentice Hall, pp. 386–397.

Read, E. & George, J. (1990), 'Dual dynamic programming for linear production/inventory systems', *Computers & Mathematics with Applications* **19**(11), 29–42.

Reeves, G. R. & Reid, R. C. (1988), 'Minimum values over the efficient set in multiple objective decision making', *European Journal of Operational Research* **36**, 334–338.

Robinson, S. M. (1991), *Extended Scenario Analysis*, Vol. 31 of *Annals of Operations Research*, Scientific Publishing Company, pp. 385–398.

Rockafellar, R. & Wets, R. J.-B. (1991), 'Scenarios and policy aggregation in optimization under uncertainty', *Mathematics of Operations Research* **16**(1), 119–147.

Rommelfanger, H. (1996), 'Fuzzy linear programming and applications', *European Journal of Operational Research* **92**, 512–527.

Rosen, J. & Maier, R. (1990), *Parallel Solution of Large-scale, Block-Angular Linear Programs*, Vol. 22 of *Annals of Operations Research*, Scientific Publishing Company, pp. 23–41.

Rosenhead, J. (1989), Robustness analysis: keeping your options open, *in* 'Rational Analysis for a Problematic World', John Wiley & Sons, chapter 8, pp. 193–218.

Schnaars, S. P. (1987), 'How to develop and use scenarios', *Long Range Planning* **20**(1), 105–114.

Sengupta, J. K. (1972), *Stochastic Programming*, North Holland, Amsterdam.

Solanki, R. S., Appino, P. A. & Cohon, J. L. (1993), 'Approximating the noninferior set in multiobjective linear programming problems', *European Journal of Operational Research* **68**, 356–373.

Steuer, R. E. (1986), *Multiple Criteria Optimization: Theory, Computation, and Application*, John Wiley & Sons.

Steuer, R. E. & Choo, E. (1983), 'An interactive weighted tchebycheff procedure for multiple objective programming', *Mathematical Programming* **26**, 326–344.

Tanaka, H., Ichihashi, H. & Asai, K. (1984), 'A formulation of linear programming problems based on comparison of fuzzy numbers', *Control and Cybernetics* **13**, 185–194.

Teghem, J. & Kunsch, F. (1986), 'A survey of techniques for finding efficient solutions to multi-objective integer linear programming', *Asia-Pacific Journal of Operational Research* **3**, 95–108.

Van Slyke, R. & Wets, R. J.-B. (1969), 'L-shaped linear programs with application to optimal control and stochastic programming', *SIAM Journal on Applied Mathematics* **17**, 638–663.

Wack, P. (1985*a*), 'Scenarios: shooting the rapids', *Harvard Business Review* **63**(6), 139–150.

Wack, P. (1985*b*), 'Scenarios: uncharted waters ahead', *Harvard Business Review* **63**(5), 73–89.

Wets, R. J.-B. (1974), 'Stochastic programs with fixed recourse: the equivalent deterministic program', *SIAM Review* **16**, 309–339.

Wets, R. J.-B. (1983), 'Solving stochastic programs with simple recourse', *Stochastics* **10**, 219–242.

Wets, R. J.-B. (1988), Large scale linear programming techniques. numerical techniques for stochastic optimization, *in* Y. Ermoliev & R. J.-B. Wets, eds, 'Numerical Techniques for Stochastic Optimization', Springer-Verlag, New York, chapter 3.

Wets, R. J.-B. (1989), The aggregation principle in scenario analysis and stochastic optimization, *in* S. Wallace, ed., 'Algorithms and Modelling in Mathematical Programming', Springer-Verlag, Berlin, pp. 91–113.

Zimmerman, H.-J. (1975), 'Optimale entscheidungenb bei unscharfen problemeschreibungen', *Zeitschrift für Betriebswirtschaftliche Forschung* **27**, 785–795.

Zionts, S. & Wallenius, J. (1983), 'An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions', *Management Science* **29**(5), 519–529.

Appendix A

Appendices for Chapter 6

A.1 The Output for the Louveaux and Smeers Example of Section 6.2

A.1.1 List of All Extreme Points Found

Solution Number		Objective Values			Scenario Weights			Plant Size			
		High	Average	Low	High	Average	Low	P1	P2	P3	P4
v1	*	58.00	29.60	-7.30	1.000	0.000	0.000	0.00	3.00	7.00	2.00
v2	*	48.00	44.00	15.60	0.000	1.000	0.000	1.00	2.00	5.00	2.00
v3	*	41.00	37.00	30.00	0.000	0.000	1.000	3.00	2.00	3.00	0.00
v4		49.00	26.64	23.90	0.906	-0.040	0.421	1.00	1.00	6.00	0.00
v5		36.90	44.00	-8.03	-0.906	0.040	-0.421	0.00	3.00	5.00	2.00
v6	*	49.00	39.80	23.90	0.793	0.300	0.531	1.00	1.00	6.00	0.00
v7		36.90	26.64	-8.03	-0.543	-0.807	-0.233	0.00	0.00	5.27	0.00
v8		55.00	41.60	-8.03	0.551	0.794	-0.259	1.00	2.00	7.00	0.00
v9		36.90	26.64	30.00	-0.685	-0.681	0.259	3.00	1.29	3.00	0.71
v10		36.90	44.00	15.60	-0.443	0.872	0.208	0.00	3.00	5.00	2.00
v11	*	51.00	38.60	18.70	0.961	0.000	0.277	0.00	1.00	7.00	0.00
v12	*	53.00	42.80	9.90	0.811	0.556	0.184	1.00	3.00	6.00	0.00
v13	*	45.00	41.00	27.60	0.029	0.893	0.449	1.00	2.00	5.00	0.00
v14		41.00	26.64	30.00	0.443	-0.175	0.879	3.00	2.00	3.00	0.00
v15		36.90	41.00	27.60	-0.850	0.336	0.406	0.00	3.00	5.00	0.00
v16	*	47.00	39.80	26.40	0.679	0.000	0.734	0.00	2.00	6.00	0.00
v17		58.00	26.64	-8.03	0.141	-0.989	-0.053	0.24	2.76	7.00	2.00
v18	*	55.00	41.60	4.70	0.965	0.247	0.091	0.00	3.00	7.00	0.00
v19	*	47.00	41.00	25.10	0.537	0.777	0.329	2.00	1.00	5.00	0.00

Table A.1: The Extreme Points in the Inner Bounding Polytope
This table lists the extreme points that define the inner bounding polytope. The noninferior points, which define the approximation to the noninferior set, are marked with an *.

A.1.2 List of All Solutions Found

This section lists the output produced by the XNISE routine. The first line shows the name of the extreme point, the weights used to find the solution, the value of the weighted objective function, and the building cost. The second line shows the individual scenario objective values, and the size of each plant type to be built. The remaining lines show the despatch under each scenario, that is, the recourse decisions.

```
v1  wgt: (1.0000, 0.0000, 0.0000)  Z:  58.00    Build Cost: 145.00
      z: (58.00,  29.60,  -7.30)     x: (0.00, 3.00, 7.00, 2.00)
                y(hi )                    y(mid)                    y(low)
         p1   p2   p3   p4       p1   p2   p3   p4       p1   p2   p3   p4
base   0.00 0.00 7.00 0.00     0.00 0.00 5.00 0.00     0.00 0.00 3.00 0.00
mid    0.00 3.00 0.00 0.00     0.00 1.00 2.00 0.00     0.00 0.00 3.00 0.00
peak   0.00 0.00 0.00 2.00     0.00 2.00 0.00 0.00     0.00 1.00 1.00 0.00

v2  wgt: (0.0000, 1.0000, 0.0000)  Z:  44.00    Build Cost: 116.00
      z: (48.00,  44.00,  15.60)     x: (1.00, 2.00, 5.00, 2.00)
                y(hi )                    y(mid)                    y(low)
         p1   p2   p3   p4       p1   p2   p3   p4       p1   p2   p3   p4
base   1.00 1.00 5.00 0.00     0.00 0.00 5.00 0.00     0.00 0.00 3.00 0.00
mid    0.00 1.00 0.00 0.00     1.00 2.00 0.00 0.00     1.00 0.00 2.00 0.00
peak   0.00 0.00 0.00 2.00     0.00 0.00 0.00 2.00     0.00 2.00 0.00 0.00

v3  wgt: (0.0000, 0.0000, 1.0000)  Z:  30.00    Build Cost: 92.00
      z: (41.00,  37.00,  30.00)     x: (3.00, 2.00, 3.00, 0.00)
                y(hi )                    y(mid)                    y(low)
         p1   p2   p3   p4       p1   p2   p3   p4       p1   p2   p3   p4
base   3.00 1.00 3.00 0.00     2.00 0.00 3.00 0.00     0.00 0.00 3.00 0.00
mid    0.00 1.00 0.00 0.00     1.00 2.00 0.00 0.00     3.00 0.00 0.00 0.00
peak   0.00 0.00 0.00 0.00     0.00 0.00 0.00 0.00     0.00 2.00 0.00 0.00
```

```
v4  wgt: (0.9062,-0.0402, 0.4210)  Z:  53.39    Build Cost: 113.00
     z: (49.00,  26.64,  23.90)     x: (1.00, 1.00, 6.00, 0.00)
              y(hi )                    y(mid)                    y(low)
        p1   p2   p3   p4       p1   p2   p3   p4       p1   p2   p3   p4
base   1.00 0.00 6.00 0.00     0.00 0.00 5.00 0.00     0.00 0.00 3.00 0.00
mid    0.00 1.00 0.00 0.00     1.00 0.00 0.25 0.00     0.00 0.00 3.00 0.00
peak   0.00 0.00 0.00 0.00     0.00 1.00 0.00 0.00     1.00 1.00 0.00 0.00

v5  wgt: (-0.9062, 0.0402,-0.4210)  Z: -28.29    Build Cost: 124.00
     z: (36.90,  44.00,  -8.03)     x: (3.00, 2.00, 5.00, 0.00)
              y(hi )                    y(mid)                    y(low)
        p1   p2   p3   p4       p1   p2   p3   p4       p1   p2   p3   p4
base   2.71 0.00 4.29 0.00     0.00 0.00 5.00 0.00     1.00 0.00 2.00 0.00
mid    0.29 2.00 0.00 0.00     3.00 0.00 0.00 0.00     0.00 0.00 3.00 0.00
peak   0.00 0.00 0.00 0.00     0.00 2.00 0.00 0.00     0.51 0.00 0.00 0.00

v6  wgt: (0.7927, 0.2995, 0.5309)  Z:  63.45    Build Cost: 113.00
     z: (49.00,  39.80,  23.90)     x: (1.00, 1.00, 6.00, 0.00)
              y(hi )                    y(mid)                    y(low)
        p1   p2   p3   p4       p1   p2   p3   p4       p1   p2   p3   p4
base   1.00 0.00 6.00 0.00     0.00 0.00 5.00 0.00     0.00 0.00 3.00 0.00
mid    0.00 1.00 0.00 0.00     1.00 1.00 1.00 0.00     0.00 0.00 3.00 0.00
peak   0.00 0.00 0.00 0.00     0.00 0.00 0.00 0.00     1.00 1.00 0.00 0.00

v7  wgt: (-0.5427,-0.8070,-0.2331)  Z: -39.65    Build Cost: 79.80
     z: (36.90,  26.64,  -8.03)     x: (3.18, 0.00, 3.00, 0.00)
              y(hi )                    y(mid)                    y(low)
        p1   p2   p3   p4       p1   p2   p3   p4       p1   p2   p3   p4
base   3.18 0.00 3.00 0.00     2.34 0.00 2.66 0.00     0.00 0.00 3.00 0.00
mid    0.00 0.00 0.00 0.00     0.84 0.00 0.00 0.00     0.23 0.00 0.00 0.00
peak   0.00 0.00 0.00 0.00     0.00 0.00 0.00 0.00     0.00 0.00 0.00 0.00
```

v8 wgt: (0.5506, 0.7937,-0.2586) Z: 65.38 Build Cost: 136.00
z: (55.00, 41.60, -8.03) x: (1.00, 2.00, 7.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	0.00	0.00	7.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	1.00	2.00	0.00	0.00	1.00	0.00	2.00	0.00	0.00	0.00	3.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00	0.00	0.87	0.00

v9 wgt: (-0.6845,-0.6813, 0.2594) Z: -35.63 Build Cost: 91.20
z: (36.90, 26.64, 30.00) x: (3.00, 1.20, 3.00, 0.80)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	3.00	1.00	3.00	0.00	2.17	0.00	2.83	0.00	0.00	0.00	3.00	0.00
mid	0.00	0.00	0.00	0.80	0.83	0.00	0.00	0.00	3.00	0.00	0.00	0.00
peak	0.00	0.20	0.00	0.00	0.00	1.20	0.00	0.00	0.00	1.20	0.00	0.80

v10 wgt: (-0.4434, 0.8718, 0.2083) Z: 25.25 Build Cost: 116.00
z: (36.90, 44.00, 15.60) x: (1.00, 2.00, 5.00, 2.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	1.00	0.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	2.00	0.00	0.00	1.00	2.00	0.00	0.00	1.00	0.00	2.00	0.00
peak	0.00	0.00	0.00	0.65	0.00	0.00	0.00	2.00	0.00	2.00	0.00	0.00

v11 wgt: (0.9608, 0.0000, 0.2772) Z: 54.18 Build Cost: 119.00
z: (51.00, 38.60, 18.70) x: (0.00, 1.00, 7.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	0.00	0.00	7.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	1.00	0.00	0.00	0.00	1.00	2.00	0.00	0.00	0.00	3.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00

v12 wgt: (0.8106, 0.5560, 0.1837) Z: 68.58 Build Cost: 127.00
z: (53.00, 42.80, 9.90) x: (1.00, 3.00, 6.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	1.00	0.00	6.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	3.00	0.00	0.00	1.00	1.00	1.00	0.00	0.00	0.00	3.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	1.00	1.00	0.00	0.00

v13 wgt: (0.0293, 0.8933, 0.4485) Z: 50.32 Build Cost: 104.00
z: (45.00, 41.00, 27.60) x: (1.00, 2.00, 5.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	1.00	1.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	1.00	0.00	0.00	1.00	2.00	0.00	0.00	1.00	0.00	2.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00

v14 wgt: (0.4432,-0.1754, 0.8791) Z: 39.87 Build Cost: 92.00
z: (41.00, 26.64, 30.00) x: (3.00, 2.00, 3.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	3.00	1.00	3.00	0.00	2.47	0.00	2.53	0.00	0.00	0.00	3.00	0.00
mid	0.00	1.00	0.00	0.00	0.53	0.00	0.00	0.00	3.00	0.00	0.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00	2.00	0.00	0.00

v15 wgt: (-0.8499, 0.3364, 0.4055) Z: -6.38 Build Cost: 104.00
z: (36.90, 41.00, 27.60) x: (1.00, 2.00, 5.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	1.00	0.00	4.02	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	0.00	0.98	0.00	1.00	2.00	0.00	0.00	1.00	0.00	2.00	0.00
peak	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00

v16 wgt: (0.6790, 0.0000, 0.7341) Z: 51.30 Build Cost: 110.00
z: (47.00, 39.80, 26.40) x: (0.00, 2.00, 6.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	0.00	1.00	6.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	1.00	0.00	0.00	0.00	2.00	1.00	0.00	0.00	0.00	3.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00

v17 wgt: (0.1405,-0.9886,-0.0533) Z: -17.76 Build Cost: 145.88
z: (58.00, 26.64, -8.03) x: (0.29, 2.71, 7.00, 2.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	0.00	0.00	7.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.29	2.71	0.00	0.00	0.29	0.89	1.82	0.00	0.00	0.00	3.00	0.00
peak	0.00	0.00	0.00	2.00	0.00	1.82	0.00	0.00	0.29	0.71	1.00	0.00

v18 wgt: (0.9648, 0.2467, 0.0911) Z: 63.76 Build Cost: 133.00
z: (55.00, 41.60, 4.70) x: (0.00, 3.00, 7.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	0.00	0.00	7.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	3.00	0.00	0.00	0.00	1.00	2.00	0.00	0.00	0.00	3.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00	1.00	1.00	0.00

v19 wgt: (0.5370, 0.7770, 0.3285) Z: 65.34 Build Cost: 107.00
z: (47.00, 41.00, 25.10) x: (2.00, 1.00, 5.00, 0.00)

	y(hi)				y(mid)				y(low)			
	p1	p2	p3	p4	p1	p2	p3	p4	p1	p2	p3	p4
base	2.00	0.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	3.00	0.00
mid	0.00	1.00	0.00	0.00	2.00	1.00	0.00	0.00	1.00	0.00	2.00	0.00
peak	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00

A.2 The Output for the APLP1 Example of Section 6.3

A.2.1 List of All Extreme Points Found

Solutions	Objective Values			Plant Sizes		Plant
	S1	S2	S3	G1	G2	Cost
v1 *	24088	30988	20894	0	2875	7188
v2 *	32350	22791	21711	3778	0	15111
v3 *	28461	25141	18322	2111	1111	11222
v4	25678	26708	23882	1333	2750	12208
v5	35585	34086	18322	2111	1111	11222
v6	35585	34086	23882	1904	1465	11280
v7 *	26631	25804	19215	1769	1769	11500
v8	35585	25141	18322	2111	1111	11222
v9	24088	34086	23882	0	2875	7188
v10	25844	34086	18622	1111	2111	9722
v11	35585	22791	23882	3778	0	15111
v12	28419	24059	23882	3167	1375	16104
v13 *	32350	23419	19489	3222	0	12889
v14 *	25844	27739	18622	1111	2111	9722
v15 *	29336	23781	21642	3222	861	15042
v16	24088	30988	23882	0	2875	7188
v17	24088	34086	20894	0	2875	7188
v18	35585	23419	19489	3222	0	12889
v19 *	25557	26899	20899	1278	2875	12299
v20 *	26905	25503	23445	2400	2750	16475
v21 *	28419	24059	22729	3167	1375	16106
v22 *	25678	26708	20708	1333	2750	12208

Table A.2: The Extreme Points Found While Finding the Approximation
The coordinates in criterion space of the noninferior extreme points in the approximation, and the plant sizes, and construction costs of their stage one decisions. The noninferior points, which define the approximation to the noninferior set, are marked with an *. Some stage one solutions found with weighting vectors that are not strictly positive, are the same as stage one solutions already found. For example, v9, v16 and v17 are all the same solution as v1. However the criterion vectors, and the recourse decisions, are different because of the negative weights.

A.2.2 List of All Solutions Found

This section lists the output produced by the XNISE routine. The first line shows the name of the extreme point, the individual scenario objective values, the weights used to find the solution, and the size of each plant type to be built. The remaining lines show the despatch under each scenario, that is, the recourse decisions.

```
v1    z: (24087.50, 30987.50, 20893.75)     x: (    0.00, 2875.00)
    wgt: (1.0000,0.0000,0.0000)
            y(lowg1)                  y(lowg2)                  y(high)
      g1       g2      buy      g1       g2      buy      g1       g2      buy
m1    0.00     0.00 1100.00     0.00     0.00 1100.00     0.00   687.50  312.50
m2    0.00  1200.00    0.00     0.00    50.00 1150.00     0.00   900.00    0.00
m3    0.00  1100.00    0.00     0.00  1100.00    0.00     0.00  1000.00    0.00

v2    z: (32350.00, 22791.11, 21711.11)     x: ( 3777.78,    0.00)
    wgt: (0.0000,1.0000,0.0000)
            y(lowg1)                  y(lowg2)                  y(high)
      g1       g2      buy      g1       g2      buy      g1       g2      buy
m1    0.00     0.00 1100.00  1100.00     0.00    0.00  1000.00    0.00    0.00
m2  788.89     0.00  411.11  1200.00     0.00    0.00   900.00    0.00    0.00
m3 1100.00     0.00    0.00  1100.00     0.00    0.00  1000.00    0.00    0.00

v3    z: (28461.11, 25141.11, 18322.22)     x: ( 2111.11, 1111.11)
    wgt: (0.0000,0.0000,1.0000)
            y(lowg1)                  y(lowg2)                  y(high)
      g1       g2      buy      g1       g2      buy      g1       g2      buy
m1    0.00     0.00 1100.00    44.44     0.00 1055.56  1000.00    0.00    0.00
m2  844.44     0.00  355.56  1200.00     0.00    0.00   900.00   -0.00    0.00
m3  211.11   888.89    0.00   655.56   444.44    0.00     0.00 1000.00    0.00

v4    z: (25678.33, 26708.33, 23882.22)     x: ( 1333.33, 2750.00)
    wgt: (0.6814,0.6542,-0.3283)
            y(lowg1)                  y(lowg2)                  y(high)
      g1       g2      buy      g1       g2      buy      g1       g2      buy
m1  566.67     0.00  533.33     0.00     0.00 1100.00   553.86    0.00  446.14
m2  100.00  1100.00    0.00  1200.00     0.00    0.00   646.14    0.00  253.86
m3    0.00  1100.00    0.00     0.00  1100.00    0.00     0.00 1000.00    0.00
```

```
v5    z: (35585.00, 34086.25, 18322.22)     x: ( 2111.11, 1111.11)
    wgt: (-0.6814,-0.6542,0.3283)
            y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00  884.05  215.95  1100.00    0.00    0.00  1000.00    0.00    0.00
m2 1055.56    0.00  144.44   800.00  400.00    0.00   900.00    0.00    0.00
m3    0.00    4.83 1095.17     0.00   44.44 1488.96     0.00 1000.00    0.00

v6    z: (35585.00, 34086.25, 23882.22)     x: ( 1904.49, 1464.89)
    wgt: (-0.3894,-0.1606,-0.9070)
            y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00 1100.00    0.00  1100.00    0.00    0.00  1326.37    0.00    0.00
m2  952.25    0.00  247.75   614.04  585.96    0.00   387.67    0.00  512.33
m3    0.00   71.91 1028.09     0.00    0.00 1450.41     0.00 1000.00    0.00

v7    z: (26630.77, 25803.85, 19215.38)     x: ( 1769.23, 1769.23)
    wgt: (0.8241,0.4966,0.2725)
            y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00    0.00 1100.00     0.00    0.00 1100.00  1000.00    0.00    0.00
m2  884.62  315.38    0.00  1200.00    0.00    0.00   592.31  307.69    0.00
m3    0.00 1100.00    0.00   392.31  707.69    0.00     0.00 1000.00    0.00

v8    z: (35585.00, 25141.11, 18322.22)     x: ( 2111.11, 1111.11)
    wgt: (-0.4702,0.3744,0.7992)
            y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00  884.05  215.95    44.44    0.00 1055.56  1000.00    0.00    0.00
m2 1055.56    0.00  144.44  1200.00    0.00    0.00   900.00    0.00    0.00
m3    0.00    4.83 1095.17   655.56  444.44    0.00     0.00 1000.00    0.00

v9    z: (24087.50, 34086.25, 23882.22)     x: (    0.00, 2875.00)
    wgt: (0.3365,-0.4519,-0.8262)
            y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00    0.00 1100.00     0.00    0.00 1100.00     0.00 1000.00    0.00
m2    0.00 1200.00    0.00     0.00 1082.92  117.08     0.00  334.21  565.79
m3    0.00 1100.00    0.00     0.00   67.08 1032.92     0.00 1000.00    0.00
```

```
v10   z: (25844.44, 34086.25, 18622.22)     x: ( 1111.11, 2111.11)
    wgt: (0.3287,-0.6556,0.6798)
            y(lowg1)                    y(lowg2)                    y(high)
       g1       g2      buy       g1       g2      buy       g1       g2      buy
m1   0.00     0.00  1100.00  1000.00     0.00   213.07  1000.00     0.00     0.00
m2 555.56   588.89    55.56     0.00   844.44   355.56     0.00   900.00     0.00
m3   0.00  1100.00     0.00     0.00     0.00  1100.00     0.00  1000.00     0.00

v11   z: (35585.00, 22791.11, 23882.22)     x: ( 3777.78,    0.00)
    wgt: (-0.7864,0.3261,-0.5246)
            y(lowg1)                    y(lowg2)                    y(high)
       g1       g2      buy       g1       g2      buy       g1       g2      buy
m1 689.13     0.00   410.87  1100.00     0.00     0.00  1501.53     0.00     0.00
m2 1199.76    0.00     0.24  1200.00     0.00     0.00   900.00     0.00     0.00
m3   0.00     0.00  1100.00  1100.00     0.00     0.00   998.47     0.00     1.53

v12   z: (28419.17, 24059.17, 23882.22)     x: ( 3166.67, 1375.00)
    wgt: (0.4883,0.8699,-0.0689)
            y(lowg1)                    y(lowg2)                    y(high)
       g1       g2      buy       g1       g2      buy       g1       g2      buy
m1 383.33     0.00   716.67  1100.00     0.00     0.00  1240.22     0.00     0.00
m2 1200.00    0.00     0.00  1200.00     0.00     0.00   900.00     0.00     0.00
m3   0.00  1100.00     0.00   550.00   550.00     0.00   709.78   290.22     0.00

v13   z: (32350.00, 23418.89, 19488.89)     x: ( 3222.22,    0.00)
    wgt: (-0.0000,0.8218,0.5698)
            y(lowg1)                    y(lowg2)                    y(high)
       g1       g2      buy       g1       g2      buy       g1       g2      buy
m1   0.00     0.00  1100.00   600.00     0.00   500.00  1000.00     0.00     0.00
m2 511.11     0.00   688.89  1200.00     0.00     0.00   900.00     0.00     0.00
m3 1100.00    0.00     0.00  1100.00     0.00     0.00  1000.00     0.00     0.00

v14   z: (25844.44, 27738.89, 18622.22)     x: ( 1111.11, 2111.11)
    wgt: (0.7020,0.1170,0.7025)
            y(lowg1)                    y(lowg2)                    y(high)
       g1       g2      buy       g1       g2      buy       g1       g2      buy
m1   0.00     0.00  1100.00     0.00     0.00  1100.00  1000.00     0.00     0.00
m2 555.56   588.89    55.56   744.44     0.00   455.56     0.00   900.00     0.00
m3   0.00  1100.00     0.00   255.56   844.44     0.00     0.00  1000.00     0.00
```

```
v15   z: (29336.11, 23780.56, 21641.67)     x: ( 3222.22,  861.11)
    wgt: (0.3901,0.9033,0.1787)
           y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00    0.00 1100.00   944.44    0.00  155.56 1000.00    0.00    0.00
m2 1200.00    0.00    0.00  1200.00    0.00    0.00  900.00    0.00    0.00
m3  411.11  688.89    0.00   755.56  344.44    0.00 1000.00    0.00    0.00

v16   z: (24087.50, 30987.50, 23882.22)     x: (    0.00, 2875.00)
    wgt: (0.9550,0.2059,-0.2135)
           y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00    0.00 1100.00    0.00    0.00 1100.00    0.00 1000.00    0.00
m2    0.00 1200.00    0.00    0.00   50.00 1150.00    0.00  334.21  565.79
m3    0.00 1100.00    0.00    0.00 1100.00    0.00    0.00 1000.00    0.00

v17   z: (24087.50, 34086.25, 20893.75)     x: (    0.00, 2875.00)
    wgt: (0.9071,-0.2922,0.3030)
           y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1    0.00    0.00 1100.00    0.00    0.00 1100.00    0.00  687.50  312.50
m2    0.00 1200.00    0.00    0.00 1082.92  117.08    0.00  900.00    0.00
m3    0.00 1100.00    0.00    0.00   67.08 1032.92    0.00 1000.00    0.00

v18   z: (35585.00, 23418.89, 19488.89)     x: ( 3222.22,    0.00)
    wgt: (-0.3303,0.8694,0.3675)
           y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1  411.11    0.00  752.83   600.00    0.00  500.00 1000.00    0.00    0.00
m2 1200.00    0.00    0.00  1200.00    0.00    0.00  900.00    0.00    0.00
m3    0.00    0.00 1100.00  1100.00    0.00    0.00 1000.00    0.00    0.00

v19  z: (25556.94, 26898.61, 20898.61)     x: ( 1277.78, 2875.00)
    wgt: (0.9058,0.4103,0.1053)
           y(lowg1)                  y(lowg2)                  y(high)
      g1      g2      buy      g1      g2      buy      g1      g2      buy
m1  638.89    0.00  461.11    0.00    0.00 1100.00 1000.00    0.00    0.00
m2    0.00 1200.00    0.00 1150.00   50.00    0.00  150.00  750.00    0.00
m3    0.00 1100.00    0.00    0.00 1100.00    0.00    0.00 1000.00    0.00
```

v20 z: (26905.00, 25503.00, 23445.00) x: (2400.00, 2750.00)
wgt: (0.6950,0.7190,0.0025)

	y(lowg1)			y(lowg2)			y(high)		
	g1	g2	buy	g1	g2	buy	g1	g2	buy
m1	1100.00	0.00	0.00	960.00	0.00	140.00	1000.00	0.00	0.00
m2	100.00	1100.00	0.00	1200.00	0.00	0.00	900.00	0.00	0.00
m3	0.00	1100.00	0.00	0.00	1100.00	0.00	260.00	740.00	0.00

v21 z: (28419.17, 24059.17, 22729.17) x: (3166.67, 1375.00)
wgt: (1.0000,1.0000,1.0000)

	y(lowg1)			y(lowg2)			y(high)		
	g1	g2	buy	g1	g2	buy	g1	g2	buy
m1	383.33	0.00	716.67	1100.00	0.00	0.00	1000.00	0.00	0.00
m2	1200.00	0.00	0.00	1200.00	0.00	0.00	900.00	0.00	0.00
m3	0.00	1100.00	0.00	550.00	550.00	0.00	950.00	50.00	0.00

v22 z: (25678.33, 26708.33, 20708.33) x: (1333.33, 2750.00)
wgt: (1.0000,1.0000,1.0000)

	y(lowg1)			y(lowg2)			y(high)		
	g1	g2	buy	g1	g2	buy	g1	g2	buy
m1	566.67	0.00	533.33	0.00	0.00	1100.00	1000.00	0.00	0.00
m2	100.00	1100.00	0.00	1200.00	0.00	0.00	200.00	700.00	0.00
m3	0.00	1100.00	0.00	0.00	1100.00	0.00	0.00	1000.00	0.00

Appendix B

Model of the Capacity Expansion Problem of Chapter 8

B.1 The Model

```
# AMPL  model of river system    river.mod

set     S               ordered                 ; # list of scenarios
set     bin             ordered                 ; # binary stage one options
param   wgt{S}          default 1               ; # weights on the scenario objectives
param   z_min{S}        default -Infinity       ; # minimum allowed values for the scenario objectives
var     z{s in S}       >= z_min[s]             ; # objective function values for each scenario

set  season   circular                                  ; # seasons (dry & wet)
set  load     ordered                                   ; # sections of load duration curve
set  gener                                              ; # generators (all types)
set  bld_two            within bin                      ; # can be built in stage 2 (expand or build)
set  exp_only           within bin                      ; # can only be expanded in stage 2
set  bingen             within {bin,gener}              ; # relate build options to generators
set  expgen             within {bld_two,gener}          ; # relate expansion options to generators
set  hydro    ordered   within gener                    ; # hydro plants (run of river or dam)
set  thermal            within gener                    ; # thermal plants
param  size{bin}                                        ; # MW for each plant option
param  bld_cost{bin}                                    ; # building costs
param  exp_cost{bld_two}                                ; # stage two expansion costs
param  max_exp{bld_two}                                 ; # maximum generator expansions
param  max_bld{gener}                                   ; # maximum capacity which can be built
param  lead{gener}                                      ; # construction lead times
param  explead{bld_two}     default 5                   ; # expansion lead time
param  life{gener}                                      ; # productive life
param  avail{gener}                                     ; # plant availability
param  co2{thermal}                                     ; # CO2 emissions per MW-year

set  river    ordered                                   ; # sections of river
set  gorge    ordered   within river                    ; # gorge sections of the river
set  flat     ordered   within river                    ; # flat  sections of the river
set  above              within {gorge,flat}             ; # link gorges to flats above
```

```
set  damflat            within {hydro,flat}                          ; # link dams to flats below
set  damgorge           within {hydro,gorge}                         ; # link dams to gorges above
param  genf {hydro}                                                  ; # factor from cumecs to Mw
param  minrs{hydro}                                                  ; # minimum reservoir sizes (if not 0)
param  maxrs{hydro}                                                  ; # maximum reservoir sizes (cumec-weeks)
param  v_res{hydro}                                                  ; # variable building cost - reservoir
param  inflow{season,river}                                          ; # inflows into top of gorges by season
param  cumflow{w in season,r in river}:=                               # cumulative natural flows in river sections
       if  ord(r) = 1   then inflow[w,r]
                        else inflow[w,r] + cumflow[w,prev(r,river)]  ; # inflow + flow in section above
param  minflow{S,season,river}                                       ; # minimum required river flows stage 2
param  flowlb{s in S,w in season,r in river} :=
                              cumflow[w,r]*minflow[s,w,r]            ; # lower bounds on river flows (cumecs)

param  P1                                                            ; # number of stages before the chance node
param  P2                                                            ; # number of stages after the chance node
param  P := P1 + P2                                                  ; # count of all stages
set    binp     within  {bin,1..P1}                                  ; # relate build options to periods
param  yrs{1..P}                                                     ; # length of stage
param  cumyrs{p in 1..P} :=                                           # cumulative years to start of period
        if    p = 1     then      0
                        else      sum{i in 1..p-1}yrs[i]             ;
param  weeks{season}                                                 ; # length of seasons in weeks
check:  sum{s in season} weeks[s] = 52                               ; # must have whole year
param  frac{s in season} := weeks[s]/52                              ; # length of seasons as fraction of year
param  rate                                                          ; # annual discount rate
param  disc := 1/(1+rate)                                            ; # annual discount factor
param  discb {p in 1..P} :=
        if    p = 1     then 1
                        else disc^(sum {i in 1..p-1} yrs[i] )        ; # discount from beginning of stage to now
param  discm {p in 1..P} :=
        if    p = 1     then disc^(yrs[1]/2)
                        else disc^(yrs[p]/2)*discb[p]                ; # discount from middle of stage to now
```

```
param  binok{b in bin,(b,p) in binp,t in P1+1..P} :=                  # is plant built in p available in t
       if    cumyrs[t] - cumyrs[p] < sum{(b,g) in bingen}lead[g]      # check lead time against yrs between p & t
       then  0                                                       #       lead time not yet elapsed
       else  if   cumyrs[t] - cumyrs[p]
                        > sum{(b,g) in bingen} (life[g] + lead[g])   # if plant beyond productive life
             then 0                                                  #       no longer available
             else 1                                                ; # plant can be used
param  expok{b in bld_two, p in P1+1..P1+1,t in P1+1..P} :=           # is plant built in p available in t
       if    cumyrs[t] - cumyrs[p] < explead[b]                      # check lead time against yrs between p & t
       then  0                                                       #       lead time not yet elapsed
       else  if   cumyrs[t] - cumyrs[p] - explead[b]                 # if plant beyond productive life
                                  > sum{(b,g) in bingen} life[g]     # if plant beyond productive life
             then 0                                                  #       no longer available
             else 1                                                ; # plant can be used
param  bintime{b in bin,(b,p) in binp,t in P1+1..P} :=                # yrs plant built in p can run at end
       if    binok[b,p,t] = 0                                        # if not available at all,
       then  0                                                       #       then zero
       else  min( sum{(b,g) in bingen} (life[g] + lead[g])
                           + cumyrs[p] - cumyrs[t], yrs[t] )       ; # shorter of life remaining & yrs in period
param  exptime{b in bld_two, p in P1+1..P1+1,t in P1+1..P} :=         # yrs plant built in p can run at end
       if    expok[b,p,t] = 0                                        # if not available at all,
       then  0                                                       #       then zero
       else  min( sum{(b,g) in bingen} life[g] + explead[b]
                           + cumyrs[p] - cumyrs[t], yrs[t] )       ; # shorter of life remaining & yrs in period
                     # note: only applies to plant that 'die' in the last period, not plant that commissions

param  gen_cost{S,P1+1..P,gener}                                     ; # generating costs
param  demand{S,P1+1..P,season,load}                                 ; # electricity demand - stage 2
param  loadf{load}                                                   ; # load factors
param  price{S,P1+1..P,season,load}                                  ; # power prices $/MW
param  tax{S,P1+1..P}                                                ; # CO2 tax (after chance node only)


var x{bin}                  binary ;    #            >=0, <= 1          ; # generators built in stage 1
```

```
var res{hydro,1..P1}                              >=0                ; # reservoir capacity built stage1 cumec-wks
var y{S,b in bld_two}                             >=0                ; # generator expansion - stage 2
var desp{S,P1+1..P,season,gener,load}             >=0                ; # despatch of generators to load (period 2)
var reles{S,P1+1..P,season,hydro,load}            >=0                ; # release through hydro generators
var spill{S,P1+1..P,season,hydro}                 >=0                ; # spill past the dam stage 2
var flow{s in S,P1+1..P,w in season,r in river} >=flowlb[s,w,r]      ; # river flows stage 2
var store{S,P1+1..P,season,hydro}                 >=0                ; # end of season storage stage 2

maximize Z: sum{s in S} z[s]*wgt[s]                                  ; # weighted objective function

subject to

obj {s in S} : z[s] =
       sum{p in P1+1..P,w in season,g in gener,l in load}
             desp[s,p,w,g,l]*price[s,p,w,l]*loadf[l]*frac[w]*yrs[p]*discm[p]   # sell electricity stage 2
                                                                  # less:
     - sum{p in P1+1..P,w in season,g in gener,l in load}
             desp[s,p,w,g,l]*gen_cost[s,p,g]*loadf[l]*frac[w]*yrs[p]*discm[p]  # production cost stage 2
     - sum{p in P1+1..P,w in season,t in thermal,l in load}
       desp[s,p,w,t,l]*co2[t]*tax[s,p]*loadf[l]*frac[w]*yrs[p]*discm[p]/10^6   # carbon taxes
                                                               # building costs:
     - sum{(b,p) in binp} x[b]*bld_cost[b]*discb[p]                  # generators stage 1
     - sum{b in bld_two} y[s,b]*exp_cost[b]*discb[P1+1]              # generators stage 2
     - sum{p in 1..P1,h in hydro}         res[h,p]*v_res[h]*discb[p]         ; # hydro reservoir stage 1

power {s in S,p in P1+1..P,w in season,g in gener}:                          # limit output to MW built
        sum{l in load} desp[s,p,w,g,l] <= sum{(b,g) in bingen,(b,i) in binp}  #   subject to availability
                                    x[b]*size[b]*avail[g]*bintime[b,i,p]/yrs[p]  # gentime:
                                       + sum{(b,g) in bingen: b in bld_two}
                                    y[s,b]*avail[g]*exptime[b,P1+1,p]/yrs[p]      ; #   yrs in period can run
supply {s in S,p in P1+1..P,w in season,l in load}:
       sum{g in gener} desp[s,p,w,g,l] <= demand[s,p,w,l]                    ; # can't sell more than demand
release{s in S,p in P1+1..P,w in season,h in hydro,l in load}:
                               desp[s,p,w,h,l] = reles[s,p,w,h,l]*genf[h]    ; # limit output to release
```

```
expanda {s in S,b in exp_only }:                    y[s,b] <= x[b]*max_exp[b]          ; # expansion of stage 1 plant
expandb {s in S,b in bld_two:b not in exp_only }:
                                          y[s,b] <= max_exp[b] - x[b]*size[b]          ; # building of plant at stage 2
minres  {h in hydro,p in 1..P1}:
                res[h,p] >= sum{(b,h) in bingen:(b,p) in binp} x[b]*minrs[h]          ; # minimum reservoir size
maxres  {h in hydro,p in 1..P1}:
                res[h,p] <= sum{(b,h) in bingen:(b,p) in binp} x[b]*maxrs[h]          ; # maximum reservoir size
totalbld{s in S,g in gener: g not in hydro}:
                     sum{(b,g) in expgen} (x[b]*size[b] + y[s,b]) <= max_bld[g]       ; # limit total construction
dambld  {g in gorge}: sum{(h,g) in damgorge,(b,h) in bingen}  x[b] <= 1               ; # build once only at each dam site
gorges {s in S,p in P1+1..P,w in season,g in gorge}:
      flow[s,p,w,g] = inflow[w,g] + sum{(g,f) in above} flow[s,p,w,f]                ; # river flows in gorges
flats  {s in S,p in P1+1..P,w in season,f in flat}:
      flow[s,p,w,f] = sum{(h,f) in damflat}( spill[s,p,w,h] + inflow[w,f]
                    + sum{l in load} reles[s,p,w,h,l]*loadf[l] )                     ; # river flows in flats
storage {s in S,p in P1+1..P,w in season,h in hydro}:
      store[s,p,w,h] = store[s,p,prev(w,season),h]                                    # opening from last season
                       + sum{(h,g) in damgorge} flow[s,p,w,g]*weeks[w]               # flow in from gorge above
                       - sum{(h,f) in damflat } flow[s,p,w,f]*weeks[w]               ; # flow out to flat below
maxstore {s in S,p in P1+1..P,w in season,h in hydro}:
      store[s,p,w,h] <= sum{(b,h) in bingen,(b,i) in binp} res[h,i]*binok[b,i,p]     ; # limit to capacity built

include river.dat                                                                    ;

let {s in S,p in P1+1..P,w in season,l in load}                                       # convert price from $/MWh
          price[s,p,w,l] := price[s,p,w,l]*24*365/10^6                               ; #  to $m/MWyr
```

B.2 The Data

```
/* RIVER.DAT   data file for river.mod
    5 scenarios: low  : depressed economic activity, electricity demand grows slowly, low electricity prices
                 dem1 : high economic activity, rapid growth in electricity demand, high prices
                 dem2 : same as dem1 except that ABC win contract to supply electricity to XYZ
                 CO2A : carbon taxes imposed, demand falls and prices rise, XYZ does not buy power
                 CO2B : carbon taxes imposed, demand increases and prices unchanged, XYZ does not buy power
*/

data                                                                    ;
set  S        := low  dem1 dem2 co2a co2b                               ; # scenarios
set  bin      := dam11 dam21 dam31        dam12 dam22 dam32
                 gasa1 gasb1 gasc1 gasd1  gasa2 gasb2 gasc2 gasd2       # gas-fired options
                 blka1 blkb1 blkc1        blka2 blkb2 blkc2             # black coal stations
                 bwna1 bwnb1 bwnc1        bwna2 bwnb2 bwnc2             # brown coal stations
                                          wnda2 wndb2 wndc2 wndd2      ; # wind power
set  season   := wet  dry                                               ; # two seasons
set  load     := Deal Base Mid Peak                                     ; # sections of load duration curve
set  gener    := H1   H2   H3  GS  CB  CL  W                            ; # generators (all types)
set  bld_two  := gasa1 gasb1 gasc1 gasd1  gasa2 gasb2 gasc2 gasd2       # gas-fired options
                 blka1 blkb1 blkc1        blka2 blkb2 blkc2             # black coal stations
                 bwna1 bwnb1 bwnc1        bwna2 bwnb2 bwnc2             # brown coal stations
                                          wnda2 wndb2 wndc2 wndd2      ; # wind power
set  exp_only := gasa1 gasb1 gasc1 gasd1  gasa2 gasb2 gasc2 gasd2       # gas-fired options
                 blka1 blkb1 blkc1        blka2 blkb2 blkc2             # black coal stations
                 bwna1 bwnb1 bwnc1        bwna2 bwnb2 bwnc2            ; # brown coal stations
set  bingen   := dam11 H1, dam21 H2, dam31 H3,                          # hydro dams
                 dam12 H1, dam22 H2, dam32 H3,
                 gasa1 GS, gasb1 GS, gasc1 GS, gasd1 GS,                # gas-fired options
                 gasa2 GS, gasb2 GS, gasc2 GS, gasd2 GS,
                 blka1 CB, blkb1 CB, blkc1 CB,                          # black coal options
                 blka2 CB, blkb2 CB, blkc2 CB,
                 bwna1 CL, bwnb1 CL, bwnc1 CL,                          # brown coal options
```

```
                bwna2 CL, bwnb2 CL, bwnc2 CL,
                wnda2  W, wndb2  W, wndc2  W, wndd2  W            ; # wind power options
set  expgen   := gasa1 GS, gasb1 GS, gasc1 GS, gasd1 GS,           # gas-fired options
                gasa2 GS, gasb2 GS, gasc2 GS, gasd2 GS,
                blka1 CB, blkb1 CB, blkc1 CB,                      # black coal options
                blka2 CB, blkb2 CB, blkc2 CB,
                bwna1 CL, bwnb1 CL, bwnc1 CL,                      # brown coal options
                bwna2 CL, bwnb2 CL, bwnc2 CL,
                wnda2  W, wndb2  W, wndc2  W, wndd2  W            ; # wind power options
set  hydro    := H1   H2   H3                                     ; # hydro: run of river or dam at each gorge
set  thermal  :=             GS  CB  CL                           ; # thermal plants: gas, coal: black & lignite
param  size     := dam11 150, dam21 200, dam31 150,                    # hydro dams
                   dam12 150, dam22 200, dam32 150,
                   gasa1 100, gasb1 200, gasc1 300, gasd1 400,        # MW: gas-fired options
                   gasa2 100, gasb2 200, gasc2 300, gasd2 400,
                   blka1 200, blkb1 400, blkc1 600,                   # MW: black coal options
                   blka2 200, blkb2 400, blkc2 600,
                   bwna1 200, bwnb1 400, bwnc1 600,                   # MW: brown coal options
                   bwna2 200, bwnb2 400, bwnc2 600,
                   wnda2 100, wndb2 100, wndc2 100, wndd2 100       ; # MW: wind power options
param  bld_cost  := dam11 1030, dam21 1530, dam31 1100,                # hydro dams
                   dam12 1130, dam22 1680, dam32 1210,
                   gasa1  555, gasb1  890, gasc1 1225, gasd1 1560,    # $: gas-fired options
                   gasa2  610, gasb2  980, gasc2 1350, gasd2 1720,
                   blka1  980, blkb1 1750, blkc1 2530,                # $: black coal options
                   blka2 1080, blkb2 1930, blkc2 2780,
                   bwna1 1080, bwnb1 1920, bwnc1 2750,                # $: brown coal options
                   bwna2 1190, bwnb2 2110, bwnc2 3030,
                   wnda2  530, wndb2  580, wndc2  630, wndd2 650    ; # $ : wind power options
param  exp_cost  := gasa1  3.7, gasb1  3.6, gasc1  3.5, gasd1  3.5,    # $/MW: gas-fired options
                   gasa2  3.3, gasb2  3.2, gasc2  3.2, gasd2  3.1,
                   blka1  4.2, blkb1  4.4, blkc1  4.5,                # $/MW: black coal options
                   blka2  3.8, blkb2  4.0, blkc2  4.1,
                   bwna1  4.4, bwnb1  4.8, bwnc1  5.2,                # $/MW: brown coal options
```

```
                  bwna2  4.1, bwnb2  4.3, bwnc2  4.7,
                  wnda2  3.0, wndb2  3.2, wndc2  3.4, wndd2 3.5          ; # $/MW: wind power options
param  max_exp   := gasa1 100, gasb1 100, gasc1 150, gasd1 200,            # MW: gas-fired options
                  gasa2 100, gasb2 100, gasc2 150, gasd2 200,
                  blka1 100, blkb1 200, blkc1 200,                         # MW: black coal options
                  blka2 100, blkb2 200, blkc2 200,
                  bwna1 100, bwnb1 200, bwnc1 300,                         # MW: brown coal options
                  bwna2 100, bwnb2 200, bwnc2 300,
                  wnda2 200, wndb2 200, wndc2 200, wndd2 200             ; # MW: wind power options
param  max_bld   := GS 600, CB 800, CL 1200, W 800                       ; # MW: upper bounds on total built
param  lead      := H1 10, H2 10, H3 10, GS 10, CB 10, CL 10, W 5        ; # construction lead times (yrs)
param  life      := H1  35, H2  35, H3  35, GS  30, CB  30, CL  30, W 20 ; # station life (years)
param  avail  := H1 1.0, H2 1.0, H3 1.0,
                           GS .90, CB .85, CL .80, W .3                  ; # plant availability
param  co2    := GS 946, CB 2470, CL 2680                                ; # co2 emissions tonnes/MW-year

set  river    := G1 F1  G2 F2  G3 F3                                     ; # sections of river: gorges and plains
set  gorge    := G1     G2     G3                                        ; # gorge sections of the river
set  flat     :=    F1     F2     F3                                    ; # plain sections of the river
set  above    :=       G2 F1, G3 F2                                     ; # link gorges to flats above
set  damflat  := H1 F1, H2 F2, H3 F3                                     ; # link dams to flats below
set  damgorge := H1 G1, H2 G2, H3 G3                                     ; # link dams to gorges above
param  genf   :=  H1 0.8, H2 1.4, H3 0.7                                     ; # cumecs to MW
param  minrs  :=  H1 400, H2 600, H3 600                                     ; # minimum reservoir size (cumec-weeks)
param  maxrs  :=  H1 500, H2 800, H3 800                                     ; # maximum reservoir size (cumec-weeks)
param  v_res  :=  H1 .45, H2 .42, H3 .45                                     ; # variable building cost $m/cumec-week
param  inflow :             G1   F1   G2   F2   G3   F3   :=
                 wet       120    0   60    0   40    0                    # cumecs
                 dry        40    0   10    0   20    0                  ; # inflows into the top of the gorges
param  minflow:=
       [low,wet,*]     G1 .5, F1 .1, G2 .10, F2 .10, G3 .10, F3 .10       # % of average flow in the wet
       [low,dry,*]     G1 .3, F1 .1, G2 .20, F2 .20, G3 .20, F3 .20       # % of average flow in the dry
       [dem1,wet,*]    G1 .1, F1 .1, G2 .10, F2 .10, G3 .10, F3 .10
       [dem1,dry,*]    G1 .1, F1 .1, G2 .20, F2 .20, G3 .20, F3 .20
```

```
        [dem2,wet,*]    G1 .1, F1 .1, G2 .10, F2 .10, G3 .10, F3 .10
        [dem2,dry,*]    G1 .1, F1 .1, G2 .20, F2 .20, G3 .20, F3 .20
        [co2a,wet,*]    G1 .1, F1 .1, G2 .10, F2 .10, G3 .10, F3 .10
        [co2a,dry,*]    G1 .1, F1 .1, G2 .10, F2 .10, G3 .10, F3 .10
        [co2b,wet,*]    G1 .1, F1 .1, G2 .10, F2 .10, G3 .10, F3 .10
        [co2b,dry,*]    G1 .1, F1 .1, G2 .10, F2 .10, G3 .10, F3 .10      ; # minimum allowed flows by scenario

param  P1     := 2                                                        ; # two stages before chance node
param  P2     := 2                                                        ; # two stages after chance node
set    binp   := dam11 1, dam21 1, dam31 1,                                 # hydro dams
                 dam12 2, dam22 2, dam32 2,
                 gasa1 1, gasb1 1, gasc1 1, gasd1 1,                      # gas-fired options
                 gasa2 2, gasb2 2, gasc2 2, gasd2 2,
                 blka1 1, blkb1 1, blkc1 1,                               # black coal options
                 blka2 2, blkb2 2, blkc2 2,
                 bwna1 1, bwnb1 1, bwnc1 1,                               # brown coal options
                 bwna2 2, bwnb2 2, bwnc2 2,
                 wnda2 2, wndb2 2, wndc2 2, wndd2 2                       ; # wind power options
param  yrs    := 1 5, 2 5, 3 5, 4 30                                      ; # length of stage
param  weeks  := wet 22, dry 30                                           ; # length of seasons in weeks
param  rate   := 0                                                        ; # 5% discount rate

param  gen_cost:=
       [low,3,*]  H1 .18, H2 .18, H3 .18, GS .30, CB .25, CL .26, W .09
       [low,4,*]  H1 .18, H2 .18, H3 .18, GS .35, CB .30, CL .30, W .09

       [dem1,3,*] H1 .18, H2 .18, H3 .18, GS .31, CB .25, CL .26, W .09
       [dem1,4,*] H1 .18, H2 .18, H3 .18, GS .38, CB .28, CL .23, W .09

       [dem2,3,*] H1 .18, H2 .18, H3 .18, GS .31, CB .25, CL .26, W .09
       [dem2,4,*] H1 .18, H2 .18, H3 .18, GS .38, CB .28, CL .23, W .09

       [co2a,3,*] H1 .18, H2 .18, H3 .18, GS .31, CB .24, CL .24, W .09
       [co2a,4,*] H1 .18, H2 .18, H3 .18, GS .37, CB .23, CL .20, W .09
```

```
        [co2b,3,*] H1 .18, H2 .18, H3 .18, GS .31, CB .24, CL .24, W .09
        [co2b,4,*] H1 .18, H2 .18, H3 .18, GS .37, CB .23, CL .20, W .09  ; # generation costs $m/MWyear
param  demand :=
        [*,3,wet,Deal]      low   0, dem1   0, dem2 400, co2a   0, co2b   0
        [*,3,wet,Base]      low  80, dem1 484, dem2 484, co2a 365, co2b 508
        [*,3,wet,Mid ]      low  60, dem1 162, dem2 162, co2a 125, co2b 170
        [*,3,wet,Peak]      low 105, dem1 207, dem2 207, co2a 155, co2b 217
        [*,3,dry,Deal]      low   0, dem1   0, dem2 400, co2a   0, co2b   0
        [*,3,dry,Base]      low 118, dem1 522, dem2 522, co2a 390, co2b 548
        [*,3,dry,Mid ]      low  92, dem1 194, dem2 194, co2a 145, co2b 204
        [*,3,dry,Peak]      low 205, dem1 347, dem2 347, co2a 260, co2b 264

        [*,4,wet,Deal]      low   0, dem1   0, dem2 400, co2a   0, co2b   0
        [*,4,wet,Base]      low 150, dem1 555, dem2 555, co2a 416, co2b 583
        [*,4,wet,Mid ]      low  85, dem1 187, dem2 187, co2a 140, co2b 196
        [*,4,wet,Peak]      low 132, dem1 234, dem2 234, co2a 175, co2b 246
        [*,4,dry,Deal]      low   0, dem1   0, dem2 400, co2a   0, co2b   0
        [*,4,dry,Base]      low 195, dem1 600, dem2 600, co2a 450, co2b 630
        [*,4,dry,Mid ]      low 122, dem1 224, dem2 224, co2a 168, co2b 235
        [*,4,dry,Peak]      low 260, dem1 404, dem2 404, co2a 303, co2b 424                    ;

param  price :=
        [*,3,wet,Deal]      low   0, dem1   0, dem2  60, co2a   0, co2b   0
        [*,3,wet,Base]      low  60, dem1  65, dem2  65, co2a  64, co2b  72
        [*,3,wet,Mid ]      low 100, dem1 110, dem2 110, co2a 108, co2b 121
        [*,3,wet,Peak]      low 115, dem1 130, dem2 130, co2a 127, co2b 143
        [*,3,dry,Deal]      low   0, dem1   0, dem2  60, co2a   0, co2b   0
        [*,3,dry,Base]      low  65, dem1  70, dem2  70, co2a  68, co2b  77
        [*,3,dry,Mid ]      low 105, dem1 121, dem2 121, co2a 118, co2b 133
        [*,3,dry,Peak]      low 128, dem1 145, dem2 145, co2a 142, co2b 160

        [*,4,wet,Deal]      low   0, dem1   0, dem2  60, co2a   0, co2b   0
        [*,4,wet,Base]      low  70, dem1  80, dem2  80, co2a  78, co2b  88
```

```
        [*,4,wet,Mid ]      low 110, dem1 130, dem2 130, co2a 127, co2b 143
        [*,4,wet,Peak]      low 130, dem1 160, dem2 160, co2a 156, co2b 176
        [*,4,dry,Deal]      low   0, dem1   0, dem2  60, co2a   0, co2b   0
        [*,4,dry,Base]      low  75, dem1  86, dem2  86, co2a  84, co2b  95
        [*,4,dry,Mid ]      low 115, dem1 143, dem2 143, co2a 140, co2b 157
        [*,4,dry,Peak]      low 136, dem1 180, dem2 180, co2a 176, co2b 198                    ;

param  loadf := Deal 1, Base 1, Mid .6, Peak .2                          ; # load durations
param  tax (tr):        low  dem1  dem2  co2a  co2b     :=
                  3     0    0     0     100   100
                  4     0    0     0     180   180                     ; # Carbon tax $/tonne carbon
model                                                                    ;
```